IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

QUINSTREET, INC.,	)	
	)	
Plaintiff/Counterclaim Defendant,	)	
	)	C.A. No. 06-495-SLR
v.	)	
	)	
PARALLEL NETWORKS, LLC,	)	
	)	
Defendant/Counterclaim Plaintiff.	)	

DECLARATION OF ANGELA M. TERRY IN SUPPORT OF PARALLEL NETWORKS' REPLY IN SUPPORT OF ITS MOTION TO STRIKE AND ITS MEMORANDUM IN OPPOSITION TO PLAINTIFF'S CROSS-MOTION FOR LEAVE TO AMEND ITS DECLARATORY JUDGMENT COMPLAINT

I, Angela M. Terry, declare as follows:

1. I am a member of good standing of the State Bar of Illinois and an attorney at the law firm of Jenner & Block, LLP, counsel for Defendant, Parallel Networks, LLC ("Parallel Networks"). I have personal knowledge of the matters stated and could competently testify to them if I were called as a witness.

2. Attached hereto as Exhibit A is a true and correct copy of U.S. Patent No. 4,924,011 entitled "Process for Preparing Taxol" referenced in *Bristol-Myers Squibb Co. v. Rhone-Poulenc Rorer, Inc.*, 326 F.3d 1226 (Fed. Cir. 2003).

3. Attached hereto as Exhibit B is a true and correct copy of U.S. Patent No. 5,894,554 entitled "System for Managing Dynamic Web Page Generation Requests by Intercepting Request at Web Server and Routing to Page Server Thereby Releasing Web Server to Process Other Requests."

4. Attached hereto as Exhibit C is a true and correct copy of U.S. Patent No. 6,415,335 entitled "System and Method for Managing Dynamic Web Page Generation Requests."

5. Attached hereto as Exhibit D is a true and correct copy of a letter from Patrick L. Patras to David L. Doyle dated September 10, 2007.

I declare under penalty of perjury under the laws of the United States that the foregoing is true and correct and that this Declaration was executed on this 19th day of February, 2008 at Chicago, Illinois.

By: */s/ Angela M. Terry*

Angela M. Terry
Jenner & Block LLP
330 North Wabash Avenue
Chicago, Illinois 60611
Tel: (312) 840-8648
aterry@jenner.com

Attorney for Defendant
Parallel Networks, LLC

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

CERTIFICATE OF SERVICE

I, David E. Moore, hereby certify that on February 19, 2008, the attached document was electronically filed with the Clerk of the Court using CM/ECF which will send notification to the registered attorney(s) of record that the document has been filed and is available for viewing and downloading.

I further certify that on February 19, 2008, the attached document was Electronically Mailed to the following person(s):

Jeffrey L. Moyer
Anne Shea Gaza
Richards, Layton & Finger
One Rodney Square
920 N. King Street
Wilmington, DE 19899
rbeiser@vedderprice.com
rzachar@vedderprice.com
lkolman@vedderprice.com

Robert S. Beiser
David Doyle
Ludwig E. Kolman
Robert S. Rigg
Vedder, Price, Kaufman & Kammholz, P.C.
222 North LaSalle Street, Suite 2500
Chicago, IL 60601
rbeiser@vedderprice.com
ddoyle@vedderprice.com
lkolman@vedderprice.com
rrigg@vedderprice.com

Gordon C. Atkinson
Cooley Godward LLP
101 California Street, 5th Flr.
San Francisco, CA 94111
atkinsongc@cooley.com

/s/ David E. Moore
Richard L. Horwitz
David E. Moore
Potter Anderson & Corroon LLP
Hercules Plaza – Sixth Floor
1313 North Market Street
P.O. Box 951
Wilmington, DE 19899-0951
(302) 984-6000
rhorwitz@potteranderson.com
dmoore@potteranderson.com

Exhibit A

United States Patent [19]

Denis et al.

[11] **Patent Number: 4,924,011**

[45] **Date of Patent: May 8, 1990**

[54] **PROCESS FOR PREPARING TAXOL**

[75] Inventors: **Jean-Noel Denis; Andrew E. Greene,** both of Uriage; **Daniel Guenard,** Montrouge; **Francoise Gueritte-Voegelein,** Les Ulis, all of France

[73] Assignee: **Centre National De La Recherche Scientifique,** Paris, France

[21] Appl. No.: **331,807**

[22] Filed: **Apr. 3, 1989**

[30] **Foreign Application Priority Data**

Apr. 6, 1988 [FR] France 88 04512

[51] **Int. Cl.**[5] **C07D 305/14**

[52] **U.S. Cl.** **549/510;** 549/511

[58] **Field of Search** 549/510, 511

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,814,470 3/1989 Colin et al. 514/449
4,857,653 8/1989 Colin et al. .

OTHER PUBLICATIONS

Gué ritte-Voegelein et al, J. Natural Products, 50 (1), pp. 9–18, 1987.
Lataste et al, Pro. Natl. Acad. Sci. USA, 81, pp. 4090–4094, Jul. 1984.

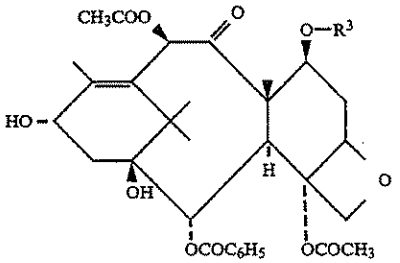
Primary Examiner—Glennon H. Hollrah

Assistant Examiner—Ba K. Trinh

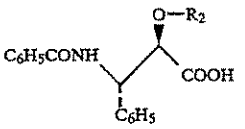
Attorney, Agent, or Firm—Stevens, Davis, Miller & Mosher

[57] **ABSTRACT**

Process for preparing taxol by the condensation of a (2R, 3S) acid of general formula (I) with a taxan derivative of general formula (II), followed by the removal of the groups R_2 and R_3 protecting the hydroxy groups.



(II)



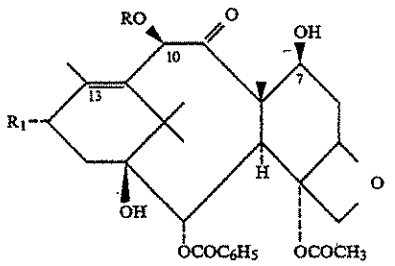
(I)

11 Claims, No Drawings

PROCESS FOR PREPARING TAXOL

The present n relates to a process for preparing taxol from a derivative of 10-deacetylbaccatine III or baccatine.

Among taxan derivatives which correspond to the general formula:



(I)

taxol is that for which R denotes an acetyl radical and R_1 denotes a (2'R,3'S) —OCO—CHOH—CH(C_6H_5)—NHCOC_6H_5 radical, 10-deacetylbaccatine III is that for which R denotes a hydrogen atom and R_1 denotes a hydroxy radical and baccatine III is that for which R denotes an acetyl radical and R_1 denotes a hydroxy radical.

Whereas taxol exhibits noteworthy properties in vitro as a promoter of tubulin polymerization and as an inhibitor of tubule depolymerization, and as a result constitutes an especially important antileukaemic and antitumour agent, 10-deacetylbaccatine III and baccatine III do not manifest these activities.

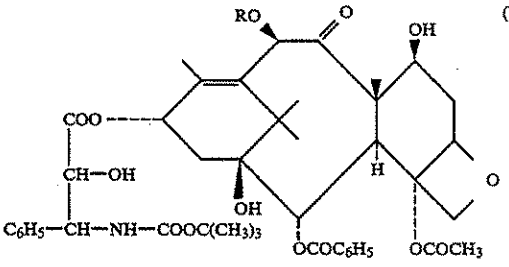
Taxol and baccatine III are extracted with difficulty and in generally low yields, of the order of 100 mg/kg in the case of taxol, from the trunk barks of different Taxus species.

Baccatine III is found in larger amounts in the wood of these different plant species.

In contrast, 10-deacetylbaccatine III is extracted much more readily and in better yields (300 mg/kg of leaves) from yew leaves.

A process enabling taxol to be prepared from 10-deacetylbaccatine III, which is readily accessible and whose production does not necessitate the total destruction of the plant species, is hence especially advantageous.

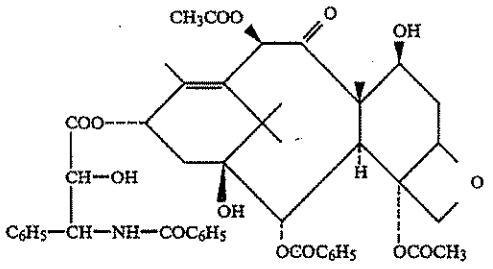
In European Patent Application EP No. 253,739, the preparation was described of taxol and 10-deacetyltaxol from a taxan derivative of general formula:



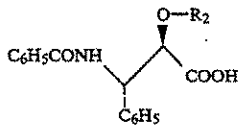
(II)

The preparation of this from baccatine III or from 10-deacetylbaccatine III, forms the subject of European Patent Application EP No. 253,738, and necessitates the intermediate separation of the diastereoisomers. As a result, it is impossible for all the baccatine III or 10-deacetylbaccatine III introduced to yield taxol having the appropriate configuration.

The present invention provides a process for preparing taxol of formula:

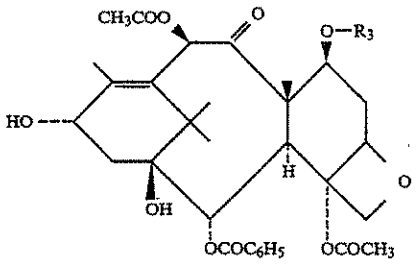


in which a (2R, 3S) 3-phenylisoserine derivative of general formula:



(III)

in which R_2 is a hydroxy-protecting group, is esterified with a taxan derivative of general formula:



(IV)

in which R_3 is a hydroxy-protecting group, and the protecting groups R_2 and R_3 are then both replaced by hydrogen.

This process may be used to produce taxol in good yield from a starting material which is easily obtained in quantity.

In the general formula (III), R_2 denotes, more especially, a methoxymethyl, 1-ethoxyethyl, benzyloxymethyl, (β-trimethylsilylethoxy)methyl, tetrahydropyranyl or 2,2,2-trichloroethoxycarbonyl radical. Preferably, R_2 is a 1-ethoxyethyl radical.

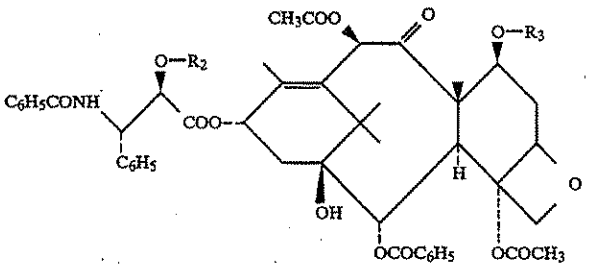
In the general formula (IV), R_3 denotes, more especially, a trialkylsilyl radical in which each alkyl portion contains 1 to 3 carbon atoms. Preferably, R_3 is a trimethylsilyl or triethylsilyl radical It is especially advantageous to use a product of general formula (IV) in which R_3 denotes a triethylsilyl radical.

In general, the esterification of the taxan derivative of general formula (IV) with the acid of general formula (III) is performed in the presence of a condensing agent, for example a carbodiimide such as dicyclohexylcarbodiimide or a reactive carbonate such as di-2-pyridyl carbonate, and an activating agent, for example a dialkylaminopyridine such as 4-dimethylaminopyridine, working in an aromatic solvent such as benzene, tolu-

ene, a xylene, ethylbenzene, isopropylbenzene or chlorobenzene at a temperature of between 60° and 90° C.

In general, an excess of acid of general formula (III) and of condensing agent (dicyclohexylcarbodiimide, di-2-pyridyl carbonate) is used, preferably 6 to 10 moles of each per mole of taxan derivative of general formula (IV), and at least one mole, and preferably 2 to 4 moles, of activating agent (4-dimethylaminopyridine) per mole of taxan derivative of general formula (IV).

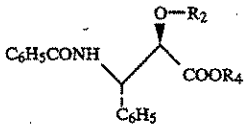
The removal of the groups protecting the (2'R,3S) ester obtained, of general formula:



(V)

in which R_2 and R_3 are defined as above, is generally accomplished by treatment in an acid medium. It is especially advantageous to use an acid such as hydrochloric acid, dissolved in an aliphatic alcohol containing 1 to 3 carbon atoms (methanol, ethanol, propanol, isopropanol) at a temperature in the region of 0° C.

The product of general formula (III) may be obtained by the saponification of a (2R, 3S) ester of general formula:



(VI)

in which R_2 is defined as above and R denotes an alkyl group containing 1 to 4 carbon atoms, and preferably methyl, by means of an inorganic base such as an alkali metal hydroxide (lithium hydroxide, sodium hydroxide) or an alkali metal carbonate or bicarbonate (sodium bicarbonate, potassium carbonate), in an aqueousalcoholic medium such as an ethanol/water or methanol/water mixture, working at a temperature of between 10° and 40° C. and preferably in the region of 25° C.

The product of general formula (VI) may be obtained under the usual conditions for preparation of ethers, and more especially according to the processes described by J. N. Denis et al., J. Org. Chem., 51, 46–50 (1986).

The product of general formula (IV) may be obtained by the action of a halotrialkylsilane on baccatine III or on 10-deacetylbaccatine III, followed, in the latter case, by the acetylation of the intermediate 7-trialkylsilyl-10-deacetylbaccatine III obtained.

In general, the reaction of the halotrialkylsilane with baccatine III or with 10-deacetylbaccatine III is performed at a temperature in the region of 20° C., working in a basic organic solvent such as pyridine or in an inert organic solvent such as chloroform or dichloroethane in the presence of a tertiary amine such as triethylamine, Hünig's base or pyridine.

The acetylation of the 7-trialkylsilyl-10-deacetylbaccatine III is generally accomplished by means of acetyl chloride, working at a temperature in the region of 0° C. in a basic organic solvent such as pyridine or in an inert organic solvent such as methylene chloride, chloroform or dichloroethane in the presence of a tertiary amine such as pyridine or Hünig's base.

The example which follows, given without implied limitation, shows how the invention can be put into practice.

EXAMPLE

42.8 mg (0.12 mmol) of N-benzoyl-O-(1-ethoxyethyl)-3-phenylisoserine in 1 cm^3 of anhydrous toluene are introduced under an argon atmosphere into a 5-cm^3 round-bottomed flask equipped with a magnetic stirrer. 25.9 mg (0.12 mmol) of di-2-pyridyl carbonate are then added. The mixture is left to react for 4 to 5 minutes, and 4.9 mg (0.04 mmol) of 4-dimethylaminopyridine and 14 mg (0.02 mmol) of 7-triethylsilylbaccatine III are then added in a single portion. The colourless and homogeneous solution is left for 3 to 4 minutes, and then heated for 10 hours at 72°–74° C. After being cooled, the reaction mixture is diluted by adding ethyl acetate. The organic solution is washed 3 times with saturated aqueous sodium bicarbonate solution, twice with water and then twice with saturated sodium chloride solution. The organic phase is dried over anhydrous sodium sulphate. After filtration and removal of the solvents under reduced pressure (20 mm of mercury; 2.7 kPa), the residue obtained is purified by analytical thin-layer chromatography on silica, eluting with an ether/methylene chloride (5:95 by volume) mixture, 4 runs being performed. 8.4 mg (0.0081 mmol) of ester of general formula (V) in which R_2 denotes a 1-ethoxyethyl radical and R_3 denotes a triethylsilyl radical is thereby obtained, in a 40% yield, in the form of a mixture of 2 epimers in the ratio 60:40, melting at 169°–173° C. after recrystallization in a methylene chloride/pentane mixture.

5.8 mg of 7-triethylsilylbaccatine III are recovered. The ester obtained has the following characteristics:

optical rotation: $[\alpha]^{24} = -33.7°$ (c=0.41; methanol)

infrared spectrum (film): 3450, 3300, 3060, 3025, 2950, 2930, 2900, 2870, 1740, 1720, 1640, 1600, 1580, 1520, 1480, 1450, 1365, 1310, 1260, 1240, 1175, 1140, 1105, 1090, 1080, 1020, 980, 945, 820 and 710 cm^{-1}

proton nuclear magnetic resonance spectrum (300 MHz; deuterated chloroform; chemical shift in ppm; coupling constants J in Hz):

0.53–0.62 (m, 6H); 0.93 (t, J=8, 9H); 1.00 (t, J=7, less abundant epimer 3H); 1.04 (t, J =7, preponderant epimer 3H); 1.18 (preponderant epimer) and 1.19 (less abundant epimer) (2s, 3H); 1.22 (s, 3H); 1.20 and 1.29 (2d, J=5.3, 3H); 1.70 (s, 3H); 1.85–1.95 (m, 1H); 2.00 and 2.01 (2d, J=1.2, 3H); 2.05-2.20 (m, 1H); 2.16 (s, 3H);

2.26–2.40 (m, 1H); 2.40 (preponderant epimer) and 2.53 (less abundant epimer) (2s, 3H); 2.46–2.59 (m, 1H); 3.04–3.44 (m, 2H); 3.81 (preponderant epimer) and 3.83 (less abundant epimer) (2d, J=7, 1H); 4.24 (less abundant epimer) and 4.26 (preponderant epimer) (2ABq, J_{AB} =8.1, $\delta_A-\delta_B$=34, 2H); 4.47 (dd, J=6.6 and 10.6, 1H); 4.64 (less abundant epimer) and 4.72 (preponderant epimer) (2d, J=2.7 and 3.7, 1H); 4.54 (less abundant epimer) and 4.80 (preponderant epimer) (2q, J=5.3, 1H); 4.94 (ps.-t, J=6.7 and 7.3, 1H); 5.68–5.76 (m, 2H); 6.24 (ps.-t, J=8 and 9, 1H); 6.44 (s, 1H); 7.08 (less abundant epimer) and 7.18 (preponderant epimer) (2d, J=8.6 and 8.1, 1H); 7.28–7.53 (m, 10H); 7.57–7.63 (m, 1H); 7.77–7.80 (m, 2H); 8.10–8.15 (m, 2H)

mass spectrum (FAB; NBA matrix): m/e=1040 (MH^+)

elemental analysis: $C_{57}H_{73}O_{15}SiN$

Calculated %	C 65.81	H 7.07	N 1.35
Found	65.57	7.34	1.62

72 mg (0.009 mmol) of the ester obtained above are introduced at 0° C. under an argon atmosphere into a 10-cm^3 round-bottomed flask equipped with a magnetic stirrer. 3.6 cm^3 of a 0.5% strength ethanolic hydrochloric acid solution, cooled beforehand to 0° C., are added. The mixture is stirred at 0° C. for 30 hours. When the reaction is complete, the reaction mixture is diluted by adding ethylacetate at 0° C., and water is then added. After settling has taken place, the separated organic phase is washed 5 times with water and twice with saturated sodium chloride solution and is then dried over anhydrous sodium sulphate. After filtration, the solvents are removed under reduced pressure (20 mm of mercury; 2.6 kPa). The residue obtained (72 mg) is purified by preparative thinlayer chromatography on silica, eluting with a dichloromethane/methanol (90:10 by volume) mixture. 54 mg (0.063 mmol) of taxol are thereby obtained.

The yield is 91%.

The taxol thereby obtained has the following characteristics:

optical rotation: $[\alpha]^{24}=-49.7°$ (c=0.36; methanol)

infrared spectrum (film): 3400, 3060, 3025, 3000, 2950, 2900, 1740, 1720, 1650, 1600, 1580, 1520, 1480, 1450, 1370, 1315, 1260, 1240, 1180, 1110, 1070, 1030, 980, 950, 905, 800 and 710 cm^{-1}

proton nuclear magnetic resonance spectrum (300 MHz; deuterated chloroform; chemical shifts in ppm; coupling constants in Hz):

1.15 (s, 3H); 1.24 (s, 3H); 1.69 (s, 3H); 1.80 (s, 3H); 1.83 (s, 1H); 1.83–1.93 (m, 1H); 2.24 (s, 3H); 2.28–2.39 (m, 2H); 2.39 (s, 3H); 2.47 (d, J=4, 1H); 2.48–2.57 (m, 1H); 3.55 (d, J=5, 1H); 3.80 (d, J=7, 1H); 4.25 (ABq, J_{AB}=8.4, $\delta_A-\delta_B$=32, 2H); 4.37–4.44 (m, 1H); 4.80 (dd, J=2.5 and 5, 1H); 4.95 (d, J=7.7, 1H); 5.68 (d, J=7, 1H); 5.79 (dd, J=2.5 and 8.8, 1H); 6.23 (t, J=9, 1H); 6.27 (s, 1H); 6.98 (d, J=8.8, 1H); 7.33–7.54 (m, 10H); 7.59–7.64 (m, 1H); 7.72–7.75 (m, 2H); 8.12–8.15 (m, 2H)

^{13}C nuclear magnetic resonance spectrum (deuterated chloroform):

9.57 (CH_3); 14.83 (CH_3); 20.83 (CH_3); 21.62 (CH_3); 22.85 (CH_3); 26.69 (CH_3); 35.65 (CH_2); 35.73 (CH_2); 43.20 (C); 45.86 (CH); 55.05 (CH); 58.67 (C); 72.20 (CH); 72.41 (CH); 73.23 (CH); 75.00 (CH); 75.58 (CH); 76.58 (CH_2); 79.10 (C); 81.21 (C); 84.42 (CH); 127.06 (CH); 128.38 (CH); 128.72 (CH); 129.04 (CH); 129.21 (C); 130.22 (CH); 131.97 (CH); 133.26 (C); 133.71 (CH); 138.03 (C); 141.98 (C); 167.04 (C); 170.37 (C); 171.22 (C); 172.73 (C); 203.62 (C)

mass spectrum (FAB; NBA matrix): m/e =854 (MH^+).

(2R, 3S)-N-Benzoyl-O-(1-ethoxyethyl)-3-phenylisoserine may be obtained in the following manner:

380 mg of N-benzoyl-O-(1-ethoxyethyl)-3-phenylisoserine methyl ester are added into a 100 cm^3 round-bottomed flask equipped with a magnetic stirrer and containing 30 cm^3 of methanol. To the solution obtained, 15 cm^3 of distilled water and 414 mg (3 mmol) of solid potassium carbonate are added. The mixture is stirred for 40 hours at 25° C. and the methanol is then evaporated off under reduced pressure. The residual aqueous phase is extracted several times with ether. The aqueous phase is acidified with 10% strength (w/v) aqueous hydrochloric acid solution and then extracted with dichloromethane. The combined organic phases are washed several times with water and then with saturated sodium chloride solution. The organic phases are dried over anhydrous magnesium sulphate. After filtration and removal of the solvent under reduced pressure, 254 mg (0.711 mmol) of N-benzoyl-O-(1-ethoxyethyl)-3-phenylisoserine are obtained, the characteristics of which are as follows:

melting point: 93°–94° C.

infrared spectrum (film): 3425, 3600–2100, 3060, 3025, 2950, 2925, 1740, 1640, 1600, 1580, 1520, 1480, 1440, 1300, 1140, 1075, 1020, 950, 920, 865, 800, 770 and 700 cm^{-1}

proton nuclear magnetic resonance spectrum (300 MHz; deuterated chloroform; chemical shifts in ppm; coupling constants J in Hz): 0.90 and 1.07 (2t, J=7, 3H); 1.24 (d, J=5.3, 3H); 2.88–2.99 and 3.24–3.45 (2m, 2H); 4.50 and 4.63 (2d, J=2.4, 1H); 4.60 and 4.81 (2q, J=5.3, 1H); 5.74–5.80 (m, 1H); 7.26–7.52 (m, 4H); 7.48–7.83 (m, 2H); 7.0–7.8 (broad s, 1H).

(2R, 3S)-N-Benzoyl-O-(1-ethoxyethyl)-3-phenylisoserine methyl ester may be prepared in the following manner:

299 mg (1 mmol) of N-benzoyl-3-phenylisoserine methyl ester, 10 cm^3 of dry dichloromethane, 25.1 mg (0.1 mmol) of pyridinium p-toluenesulphonate and 956.4 µl (721 mg; 10 mmol) of ethyl vinyl ether are introduced successively into a 25-cm^3 round-bottomed flask equipped with a magnetic stirrer. The reaction mixture is stirred for 3 hours at 25° C. When the reaction is complete, 1 drop of pyridine is added and the reaction mixture is then diluted by adding dichloromethane. The organic phase is washed twice with water and then with saturated sodium chloride solution, and dried over anhydrous sodium sulphate. After filtration and removal of the solvents under reduced pressure, 380 mg of N-benzoyl-O-(1-ethoxy- ethyl)-3-phenylisoserine methyl ester are obtained in the form of an equimolecular mixture of 2 epimers, the characteristics of which are as follows:

melting point: 124°–125° C. (after recrystallization in a dichloromethane/pentane mixture)

optical rotation: $[\alpha]^{23}=-25.9°$ (c=0.54; methanol)

infrared spectrum (film): 3350, 3060, 3025, 2980, 2940, 1740, 1635, 1600, 1580, 1530, 1490, 1435, 1380, 1340, 1320, 1275, 1242, 1198, 1175, 1150, 1080, 1030, 990, 955, 900, 800, 702 and 698 cm^{-1}

proton nuclear magnetic resonance spectrum (300 MHz; deuterated chloroform; chemical shifts in ppm; coupling constants J in Hz):

0.87 and 0.98 (2t, J=7, 3H); 1.13 and 1.22 (2d, J=5.4, 3H); 2.82 and 2.89 and 3.22–3.36 (m, 2H); 3.755 and 3.76 (2s, 3H); 4.48 and 4.60 (2d, J=2.4, 1H); 4.50 and 4.78 (2q, J=5.4, 1H); 5.64 and 5.68 (2dd, J =2.4 and 8.2, 1H); 7.18 and 7.19 (2d, J=8.2, 1H); 7.23–7.55 (m, 8H); 7.80–7.84 (m, 2H)

mass spectrum (FAB, NBA matrix): m/e =372 (MH^+)

elemental analysis: $C_{21}H_{25}O_5N$

Calculated %	C 67.90	H 6.78	N 3.77
Found	67.98	6.94	3.75

N-Benzoyl-3-phenylisoserine methyl ester may be prepared according to J.N. Denis et al., J. Org. Chem., 51. 46–50 (1986).

7-Triethylsilylbaccatine III may be prepared in one of the following ways:

(a) 544 mg (1 mmol) of 10-deacetylbaccatine III, dissolved in 50 cm^3 of anhydrous pyridine, are introduced under an argon atmosphere into a 100-cm^3 round-bottomed flask equipped with a magnetic stirrer. 3.36 cm^3 (3.014 g; 20 mmol) of triethylsilyl chloride are then added. The homogeneous, yellow reaction mixture is then stirred for 24 hours at 0° C. Ethyl acetate and water are then added. After settling has taken place, the separated aqueous phase is extracted with ethyl acetate. The combined organic phases are treated with saturated aqueous copper sulphate solution until the pyridine has been completely removed. The organic phases are washed with water and then with saturated sodium chloride solution, and then dried over anhydrous sodium sulphate. After filtration and evaporation of the solvents under reduced pressure, 2.73 g of a product are obtained, and this is purified on a silica column, eluting with a dichloromethane/methanol (99:1 by volume) mixture. 512 mg (0.778 mmol) of 10-deacetyl-7-triethylsilylbaccatine III are thereby obtained, in a 78% yield, in the form of a white solid, the characteristics of which are as follows:

melting point: 256°–257° C. (after recrystallization in a dichloromethane/pentane mixture)

optical rotation: $[\alpha]^{23}=-23.6°$ (c=0.41; methanol)

infrared spectrum (film): 3450, 2950, 2875, 1735, 1705, 1600, 1580, 1450, 1380, 1275, 1242, 1180, 1140, 1115, 1100, 1075, 1060, 1030, 1020, 1000, 990, 950, 925, 885, 860, 822, 740 and 715 cm^{-1}

proton nuclear magnetic resonance spectrum (300 MHz; deuterated chloroform; chemical shifts in ppm; coupling constants J in Hz):

0.48–0.70 (m, 6H); 0.94 (t, J=8, 9H); 1.08 (s, 6H); 1.57 (s, 1H); 1.74 (s, 3H); 1.86–1.95 (m, 1H); 2.02 (d, J=5, 1H); 2.09 (d, J=1.1, 3H); 2.25–2.35 (m, 2H); 2.28 (s, 3H); 2.42–2.52 (m, 1H); 3.95 (d, J =7, 1H); 4.24 (ABq, $J_{AB}=8.2$, $\delta_A-\delta_B=42$, 2H); 4.24 (d, J =2, 1H); 4.41 (dd, J=6.6 and 10.6, 1H); 4.85–4.90 (m, 1H); 4.95 (dd, J=1.9 and 9.6, 1H); 5.17 (d, J=2, 1H); 5.60 (d, L, J=7, 1H); 7.44–7.50 (m, 2H); 7.57–7.62 (m, 1H); 8.08–8.11 (m, 2H)

^{13}C nuclear magnetic resonance spectrum (deuterated chloroform):

5.16 (CH_2); 6.72 (CH_3); 9.92 (CH_3); 15.13 (CH_3); 19.50 (CH_3); 22.60 (CH_3); 26.86 (CH_3); 37.26 (CH_2); 38.64 (CH_2); 42.70 (C); 46.99 (CH); 57.96 (C); 67.95 (CH); 72.95 (CH); 74.68 (CH); 74.83 (CH); 76.57 (CH_2); 78.78 (C); 80.75 (C); 84.25 (CH); 128.57 (CH); 129.44 (C); 130.07 (CH); 133.57 (CH); 135.20 (C); 141.78 (C); 167.04 (C); 170.76 (C); 210.31 (C)

mass spectrum (FAB; NBA matrix): m/e=659 (MH^+)

elemental analysis: $C_{35}H_{50}O_{10}Si$

Calculated %	C 63.80	H 7.65	Si 4.26
Found	63.57	7.72	4.04

43.9 mg (0.075 mmol) of 10-deacetyl-7-triethylsilylbaccatine III, dissolved in 1.87 cm^3 of anhydrous pyridine, are introduced under an argon atmosphere into a 10-cm^3 round-bottomed flask equipped with a magnetic stirrer. The solution is cooled to 0° C. and 26.6 μl (29.4 mg; 0.375 mmol) of acetyl chloride are then added dropwise. The mixture, which becomes heterogeneous, is stirred for 20 hours at 0° C. A further 26.6 μl of acetyl chloride are added and the mixture is then stirred for 20 hours at 0° C. Ethyl acetate is added, followed by water, at 0° C. After settling has taken place, the separated aqueous phase is extracted twice with ethyl acetate. The combined organic phases are washed with saturated aqueous copper sulphate solution until the pyridine has been completely removed, then with water and then with saturated sodium chloride solution, and are finally dried over anhydrous sodium sulphate. After filtration and removal of the solvents under reduced pressure, the residue obtained (65 mg) is purified by chromatography on a silica gel column, eluting with a dichloromethane/methanol (99:1 by volume) mixture. 45 mg (0.0643 mmol) of 7-triethylsilylbaccatine III are thereby obtained in an 86% yield, the characteristics of which are as follows:

melting point: 253°–254° C. (after recrystallization in a dichloromethane/pentane mixture)

optical rotation: $[\alpha]^{23}=-48.6°$ (c=0.36; methanol)

infrared spectrum (film): 3500, 2950, 2875, 1720, 1600, 1580, 1450, 1370, 1270, 1240, 1180, 1140, 1110, 1100, 1075, 1050, 1020, 990, 970, 950, 820 740 and 710 cm^{-1}

proton nuclear magnetic resonance spectrum (300 MHz; deuterated chloroform; chemical shifts in ppm; coupling constants J in Hz):

0.52–0.65 (m, 6H); 0.93 (t, J=8, 9H); 1.05 (s, 3H); 1.20 (s, 3H); 1.61 (s, 1H); 1.68 (s, 3H); 1.83–1.92 (m, 1H); 2.01 (d, J=5, 1H); 2.17 (s, 3H); 2.19 (d, J =1.1, 3H); 2.24–2.28 (m, 2H); 2.28 (s, 3H); 2.41–2.58 (m, 1H); 3.88 (d, J=7, 1H); 4.23 (ABq, $J_{Ab}=8.1$, $\delta_A-\delta_B=45$, 2H); 4.45 (dd, J=6.6 and 10.5, 1H); 4.83 (m, 1H); 4.96 (d, J=9.6, 1H); 5.63 (d, J =7, 1H); 6.46 (s, 1H); 7.45–7.50 (m, 2H); 7.57–7.63 (m, 1H); 8.09–8.12 (m, 2H)

^{13}C nuclear magnetic resonance spectrum (deuterated chloroform): 5.28; 6.72; 9.93; 14.93; 20.06; 20.93; 22.68; 26.83; 37.24; 38.24; 42.79; 47.24; 58.67; 68.00; 72.35; 74.35; 75.77; 76.57; 78.73; 80.89; 84.22; 128.57; 129.38; 130.09; 132.78; 133.61; 143.83; 167.12; 169.33; 170.76 and 202.12

mass spectrum (FAB; NBA matrix): m/e=701 (MH^+)

elemental analysis: $C_{37}H_{52}O_{11}Si$

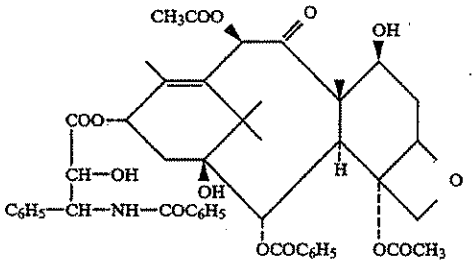
Calculated %	C 63.40	N 7.48	Si 4.01
Found	63.50	7.59	3.94

(b) 250 mg (0.4266 mmol) of baccatine III, dissolved in 8.5 cm^3 of anhydrous pyridine, are introduced under an argon atmosphere into a 25 cm^3 round-bottomed flask equipped with a magnetic stirrer. 1.43 cm^3 (1.286

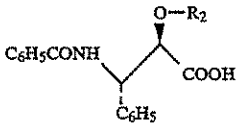
g; 8.53 mmol) of triethylsilyl chloride are then added. The mixture is stirred for 28 hours at 20° C. Ethyl acetate is added, followed by water. After settling has occurred, the aqueous phase is extracted with ethyl acetate, the combined organic phases are washed with saturated aqueous copper sulphate solution until the pyridine has been completely removed, then with water and finally with saturated aqueous sodium chloride solution. The organic phases are dried over anhydrous sodium sulphate. After filtration and evaporation of the solvents under reduced pressure, a residue (1.35 g) is obtained, and this is purified by filtration on a silica gel column, eluting with dichloromethane/methanol (99:1 by volume) mixture. 248 mg (0.354 mmol) of 7-triethylsilyl baccatine III are thereby obtained, in an 83% yield, in the form of a white solid, melting at 253°–254° C. after recrystallization in a dichloromethane/pentane mixture.

We claim:

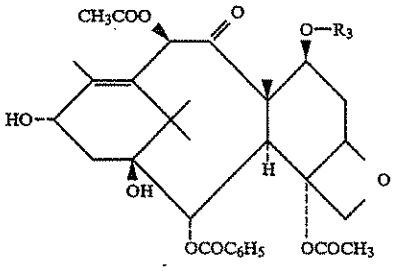
1. A process for preparing taxol of formula:



in which a (2R, 3S) 3-phenylisoserine derivative of general formula:



in which R_2 is a hydroxy-protecting group, is esterified with a taxan derivative of general formula:



in which R_3 is a hydroxy-protecting group, and the protecting groups R_2 and R_3 are then both replaced by hydrogen.

2. A process according to claim 1, in which R_2 is chosen from methoxymethyl, 1-ethoxyethyl, benzyloxymethyl, (β-trimethylsilylethoxy)methyl, tetrahydropyranyl and 2,2,2-trichloroethoxycarbonyl, and R_3 is chosen from trialkylsilyl groups in which each alkyl portion contains 1 to 3 carbon atoms.

3. A process according to claim 2, in which the esterification is performed in the presence of a condensing agent and an activating agent.

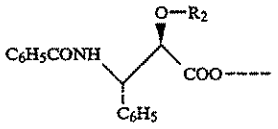
4. A process according to claim 3, in which the condensing agent is chosen from carbodiimides and reactive carbonates, and the activating agent is chosen from dialkylaminopyridines.

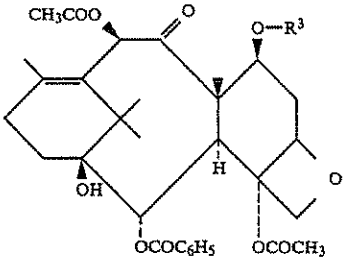
5. A process according to claim 4, in which the condensing agent is chosen from dicyclohexylcarbodiimide and di-2-pyridyl carbonate and the activating agent is 4-dimethylaminopyridine.

6. A process according to claim 1, in which the reaction is performed in an aromatic organic solvent chosen from benzene, toluene, xylenes, ethylbenzene, isopropylbenzene and chlorobenzene.

7. A process according to claim 1, in which the reaction is performed at a temperature of between 60° and 90° C.

8. A process according to claim 1, in which the replacement of the protecting groups with hydrogen in the intermediate (2'R, 3'S) ester obtained, of general formula:





in which R_2 and R_3 are as defined in claim 1, is performed in an acid medium.

9. A process according to claim 8, in which the acid medium comprises and inorganic acid dissolved in an aliphatic alcohol containing 1 to 3 carbon atoms.

10. A process according to claim 9, in which the inorganic acid is hydrochloric acid.

11. A process according to claim 8, in which the reaction is performed at a temperature in the region of 0° C.

* * * * *

Exhibit B

US005894554A

United States Patent [19]

Lowery et al.

[11] Patent Number: 5,894,554

[45] Date of Patent: Apr. 13, 1999

[54] **SYSTEM FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS BY INTERCEPTING REQUEST AT WEB SERVER AND ROUTING TO PAGE SERVER THEREBY RELEASING WEB SERVER TO PROCESS OTHER REQUESTS**

[75] Inventors: **Keith Lowery**, Richardson; **Andrew B. Levine**, Plano; **Ronald L. Howell**, Rowlett, all of Tex.

[73] Assignee: **InfoSpinner, Inc.**, Richardson, Tex.

[21] Appl. No.: **08/636,477**

[22] Filed: **Apr. 23, 1996**

[51] **Int. Cl.**[6] **G06F 13/14**; G06F 13/20

[52] **U.S. Cl.** **395/200.33**; 395/200.68; 395/200.75; 395/680; 707/10; 707/104

[58] **Field of Search** 358/400; 395/800, 395/700, 200.68, 200.75, 200.53, 680, 200.33; 707/104, 10

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,866,706	9/1989	Christophersen et al.	370/85.7
5,341,499	8/1994	Doragh	395/700
5,392,400	2/1995	Berkowitz et al.	395/200
5,404,522	4/1995	Carmon et al.	395/650
5,404,523	4/1995	DellaFera et al.	395/650
5,404,527	4/1995	Irwin et al.	395/700
5,452,460	9/1995	Distelberg et al.	395/700
5,532,838	7/1996	Barbari	358/400
5,751,956	5/1998	Kirsch	395/200.33
5,761,673	6/1998	Bookman et al.	707/104

OTHER PUBLICATIONS

"Beyond the Web: Excavating the Real World Via Mosaic", Goldberg et al. Second International WWW, Oct. 17, 1994.

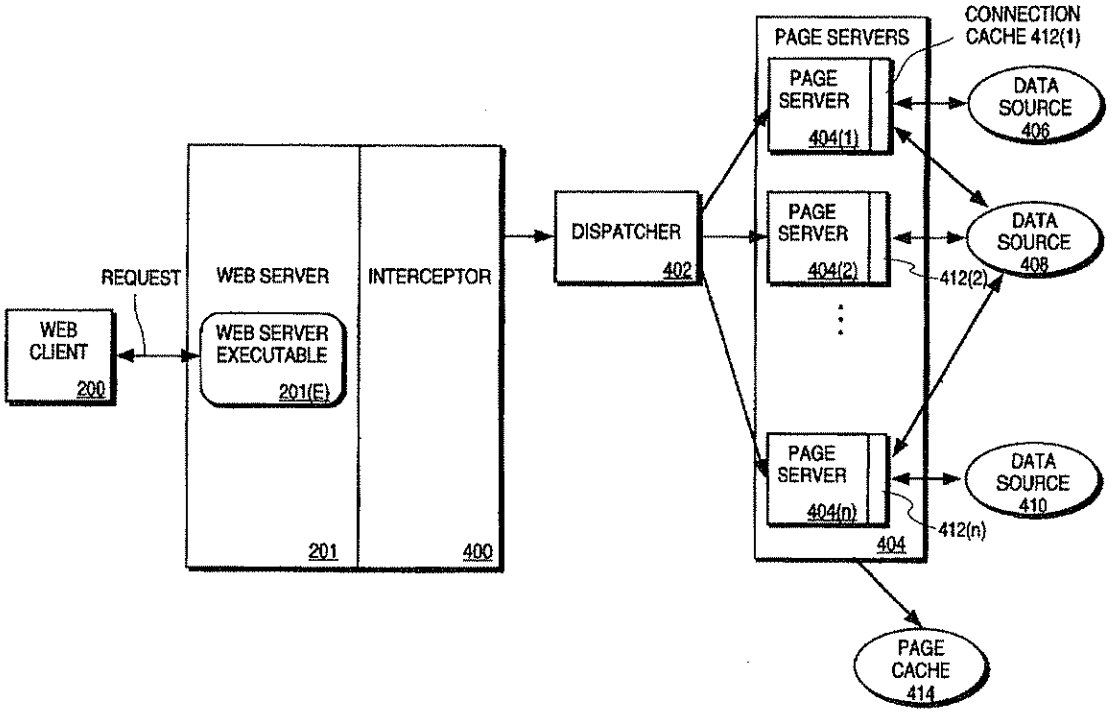
PCT International Search Report, Aug. 21, 1997.

Primary Examiner—Thomas C. Lee
Assistant Examiner—Rehana Perveen
Attorney, Agent, or Firm—Blakely, Sokoloff, Taylor & Zafman LLP

[57] **ABSTRACT**

The present invention teaches a method and apparatus for creating and managing custom Web sites. Specifically, one embodiment of the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources.

11 Claims, 5 Drawing Sheets



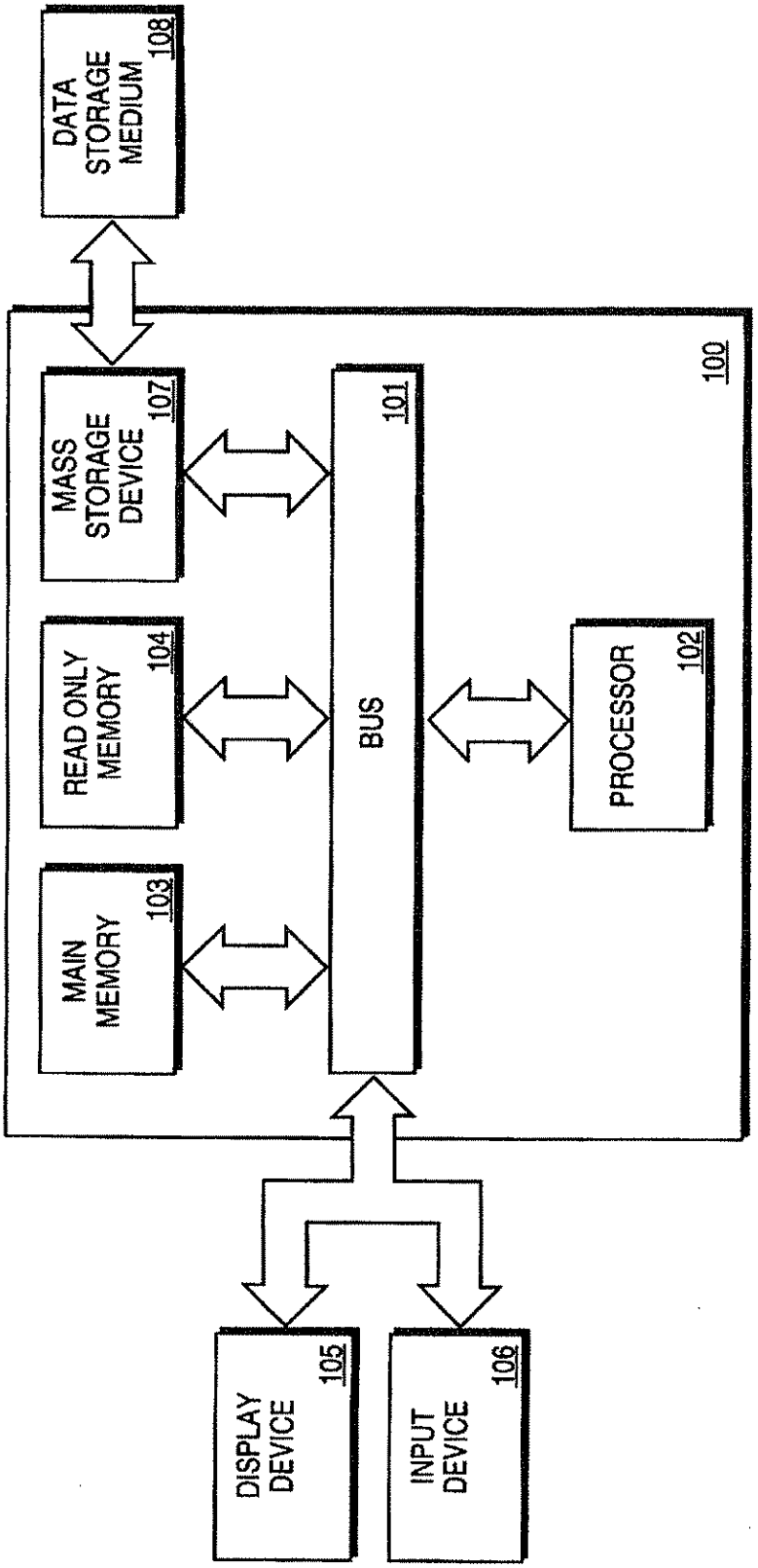

DATA STORAGE MEDIUM
108
100
MASS STORAGE DEVICE
107
BUS
101
READ ONLY MEMORY
104
PROCESSOR
102
MAIN MEMORY
103
DISPLAY DEVICE
105
INPUT DEVICE
106

FIG. 1

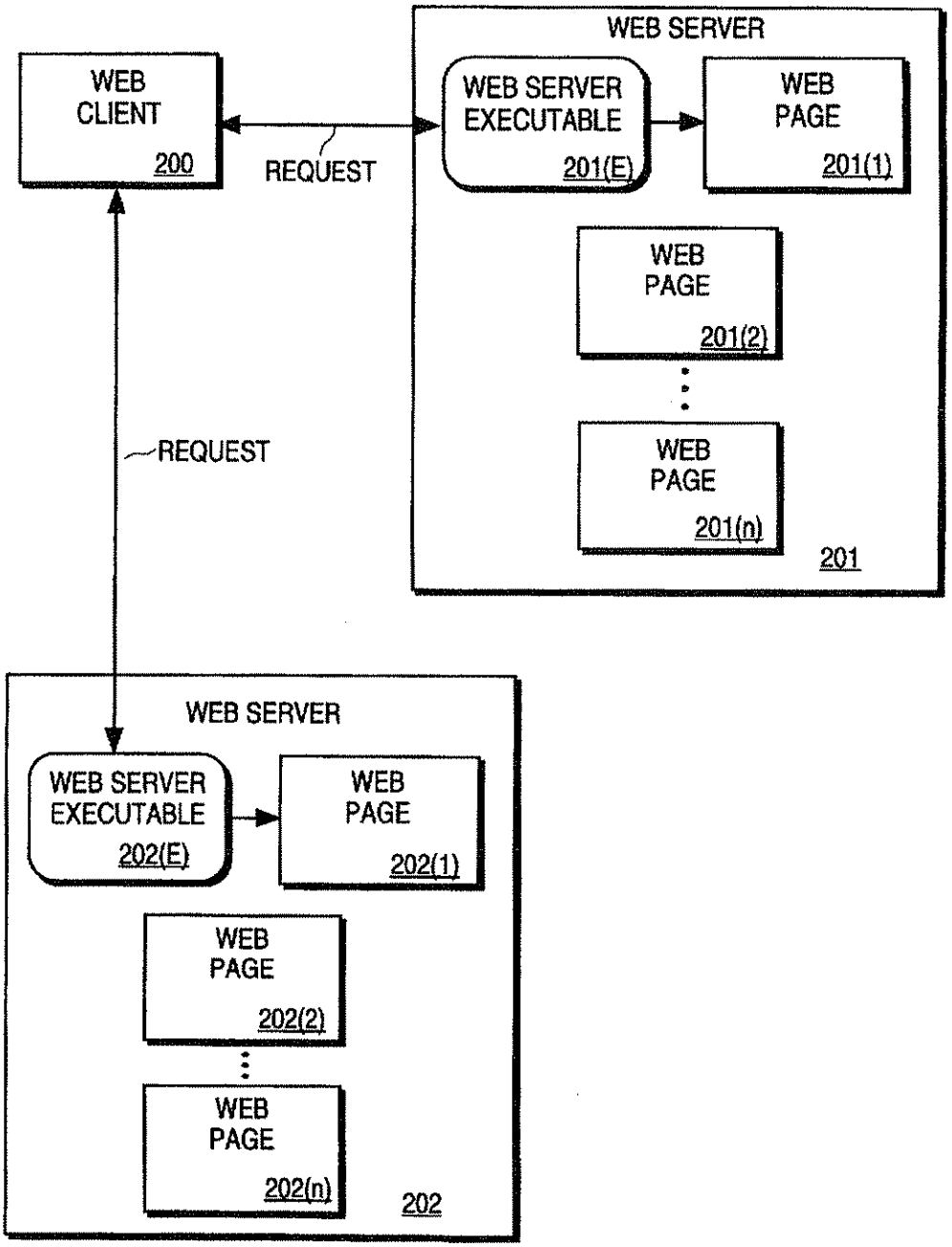


FIG. 2 (PRIOR ART)

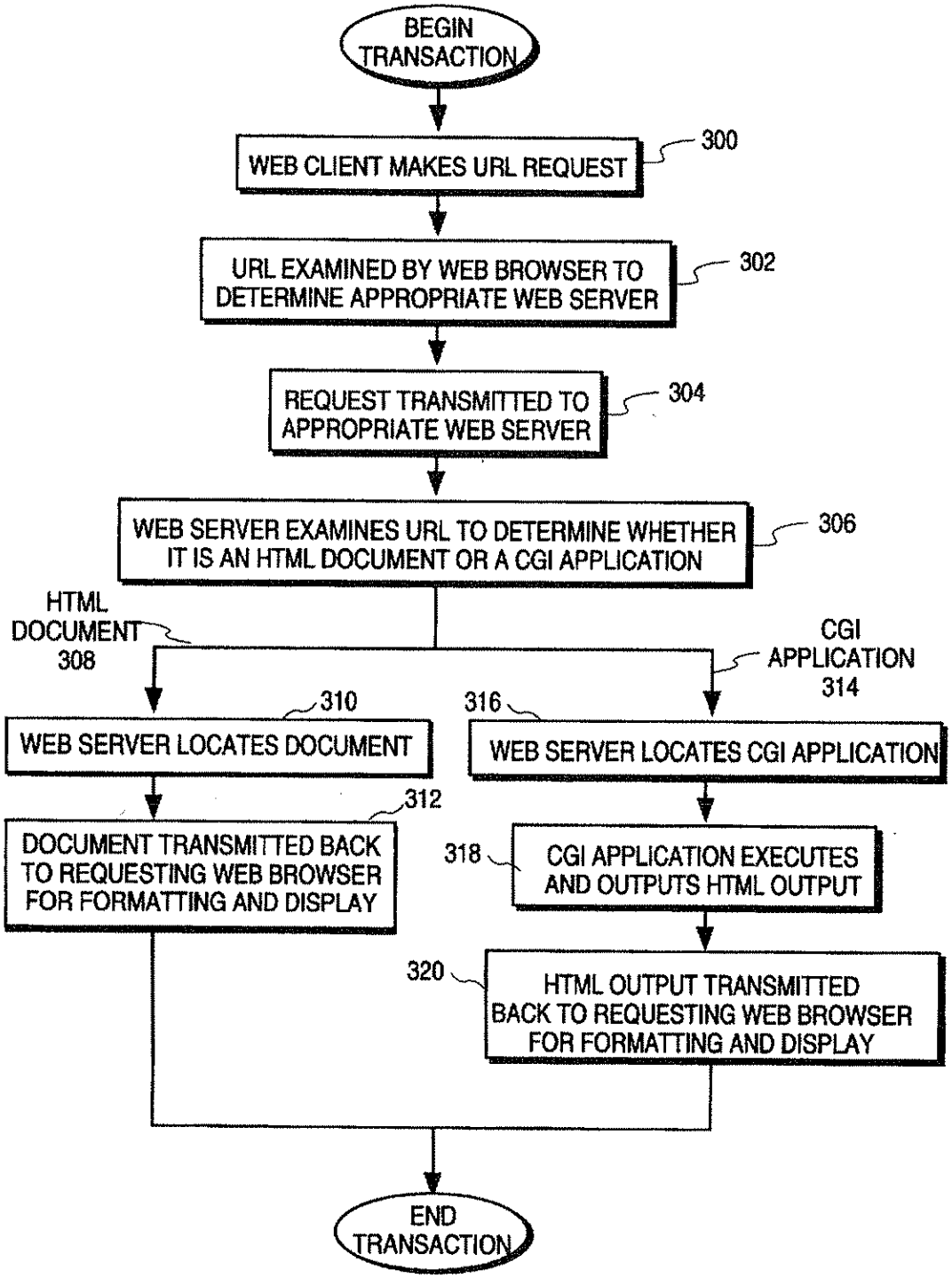


FIG. 3 (PRIOR ART)

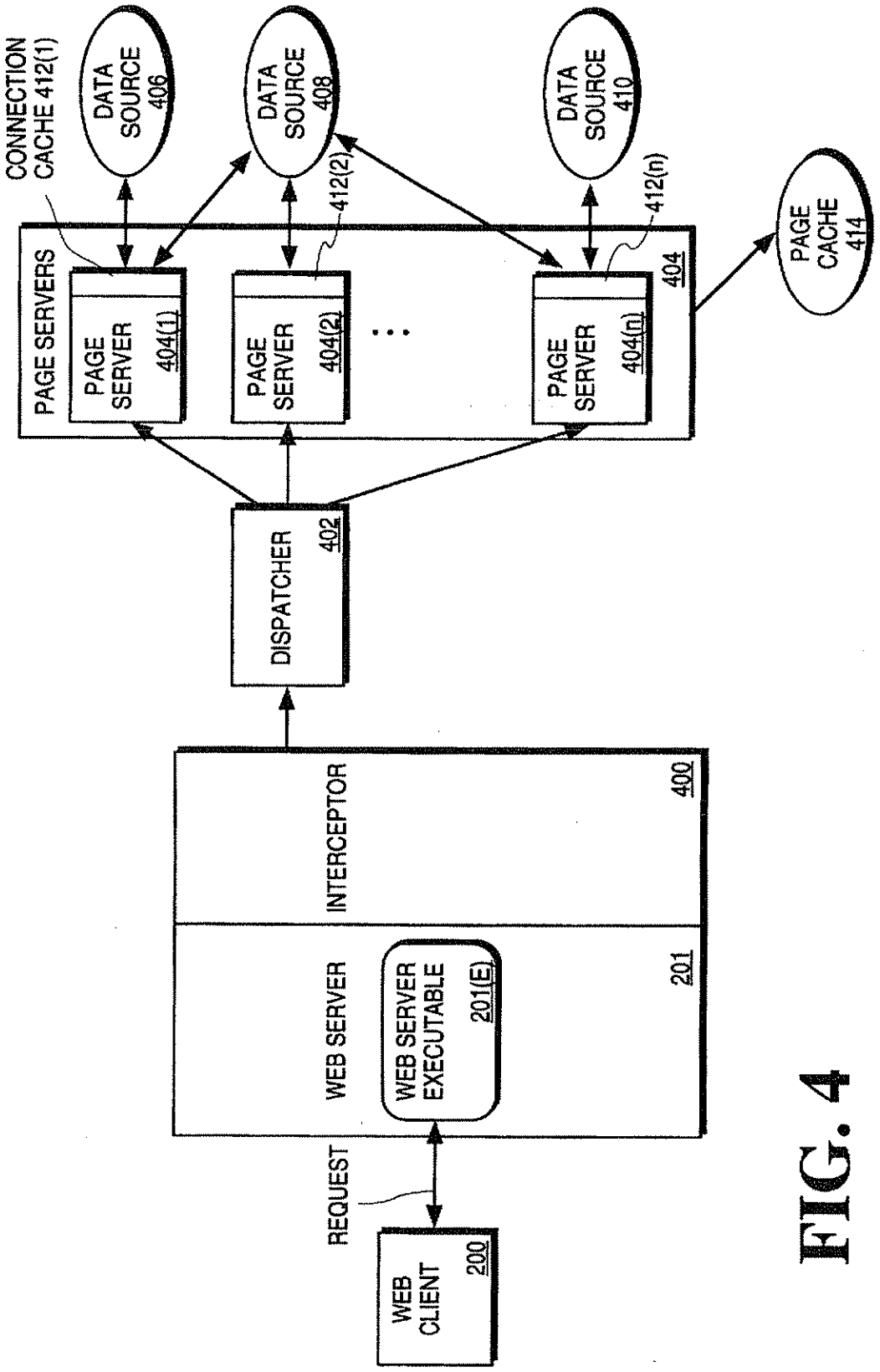

CONNECTION CACHE 412(1)
DATA SOURCE 406
DATA SOURCE 408
DATA SOURCE 410
412(2)
412(n)
PAGE CACHE 414
PAGE SERVERS
PAGE SERVER 404(1)
PAGE SERVER 404(2)
PAGE SERVER 404(n)
404
DISPATCHER 402
INTERCEPTOR 400
WEB SERVER
WEB SERVER EXECUTABLE 201(E)
201
REQUEST
WEB CLIENT 200

FIG. 4

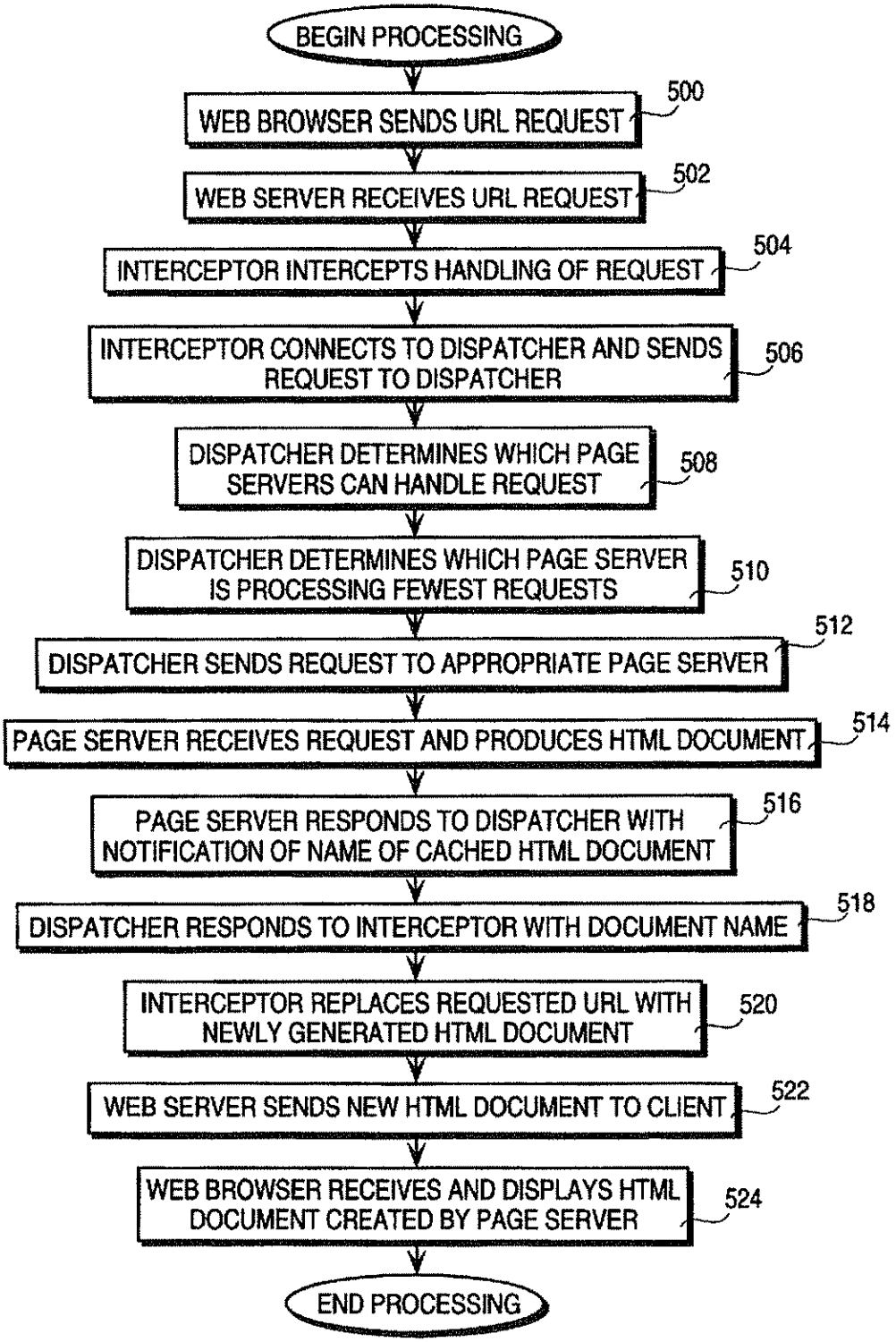

BEGIN PROCESSING
WEB BROWSER SENDS URL REQUEST
500
WEB SERVER RECEIVES URL REQUEST
502
INTERCEPTOR INTERCEPTS HANDLING OF REQUEST
504
INTERCEPTOR CONNECTS TO DISPATCHER AND SENDS REQUEST TO DISPATCHER
506
DISPATCHER DETERMINES WHICH PAGE SERVERS CAN HANDLE REQUEST
508
DISPATCHER DETERMINES WHICH PAGE SERVER IS PROCESSING FEWEST REQUESTS
510
DISPATCHER SENDS REQUEST TO APPROPRIATE PAGE SERVER
512
PAGE SERVER RECEIVES REQUEST AND PRODUCES HTML DOCUMENT
514
PAGE SERVER RESPONDS TO DISPATCHER WITH NOTIFICATION OF NAME OF CACHED HTML DOCUMENT
516
DISPATCHER RESPONDS TO INTERCEPTOR WITH DOCUMENT NAME
518
INTERCEPTOR REPLACES REQUESTED URL WITH NEWLY GENERATED HTML DOCUMENT
520
WEB SERVER SENDS NEW HTML DOCUMENT TO CLIENT
522
WEB BROWSER RECEIVES AND DISPLAYS HTML DOCUMENT CREATED BY PAGE SERVER
524
END PROCESSING

FIG. 5

SYSTEM FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS BY INTERCEPTING REQUEST AT WEB SERVER AND ROUTING TO PAGE SERVER THEREBY RELEASING WEB SERVER TO PROCESS OTHER REQUESTS

FIELD OF THE INVENTION

The present invention relates to the field of Internet technology. Specifically, the present invention relates to the creation and management of custom World Wide Web sites.

DESCRIPTION OF RELATED ART

The World Wide Web (the Web) represents all of the computers on the Internet that offer users access to information on the Internet via interactive documents or Web pages. These Web pages contain hypertext links that are used to connect any combination of graphics, audio, video and text, in a non-linear, non-sequential manner. Hypertext links are created using a special software language known as HyperText Mark-Up Language (HTML).

Once created, Web pages reside on the Web, on Web servers or Web sites. A Web site can contain numerous Web pages. Web client machines running Web browsers can access these Web pages at Web sites via a communications protocol known as HyperText Transport Protocol (HTTP). Web browsers are software interfaces that run on World Wide Web clients to allow access to Web sites via a simple user interface. A Web browser allows a Web client to request a particular Web page from a Web site by specifying a Uniform Resource Locator (URL). A URL is a Web address that identifies the Web page and its location on the Web. When the appropriate Web site receives the URL, the Web page corresponding to the requested URL is located, and if required, HTML output is generated. The HTML output is then sent via HTTP to the client for formatting on the client's screen.

Although Web pages and Web sites are extremely simple to create, the proliferation of Web sites on the Internet highlighted a number of problems. The scope and ability of a Web page designer to change the content of the Web page was limited by the static nature of Web pages. Once created, a Web page remained static until it was manually modified. This in turn limited the ability of Web site managers to effectively manage their Web sites.

The Common Gateway Interface (CGI) standard was developed to resolve the problem of allowing dynamic content to be included in Web pages. CGI "calls" or procedures enable applications to generate dynamically created HTML output, thus creating Web pages with dynamic content. Once created, these CGI applications do not have to be modified in order to retrieve "new" or dynamic data. Instead, when the Web page is invoked, CGI "calls" or procedures are used to dynamically retrieve the necessary data and to generate a Web page.

CGI applications also enhanced the ability of Web site administrators to manage Web sites. Administrators no longer have to constantly update static Web pages. A number of vendors have developed tools for CGI based development, to address the issue of dynamic Web page generation. Companies like Spider™ and Bluestone™, for example, have each created development tools for CGI-based Web page development. Another company, Haht Software™, has developed a Web page generation tool that uses a BASIC-like scripting language, instead of a CGI scripting language.

Tools that generate CGI applications do not, however, resolve the problem of managing numerous Web pages and requests at a Web site. For example, a single company may maintain hundreds of Web pages at their Web site. Current Web server architecture also does not allow the Web server to efficiently manage the Web page and process Web client requests. Managing these hundreds of Web pages in a coherent manner and processing all requests for access to the Web pages is thus a difficult task. Existing development tools are limited in their capabilities to facilitate dynamic Web page generation, and do not address the issue of managing Web requests or Web sites.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method and apparatus for creating and managing custom Web sites. Specifically, the present invention claims a method and apparatus for managing dynamic web page generation requests.

In one embodiment, the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources. Other embodiments also include connection caches to the one or more data sources, page caches for each page server, and custom HTML extension templates for configuring the Web page.

Other objects, features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a typical computer system in which the present invention operates.

FIG. **2** illustrates a typical prior art Web server environment.

FIG. **3** illustrates a typical prior art Web server environment in the form of a flow diagram.

FIG. **4** illustrates one embodiment of the presently claimed invention.

FIG. **5** illustrates the processing of a Web browser request in the farm of a flow diagram, according to one embodiment of the presently claimed invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to a method and apparatus for creating and managing custom Web sites. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent to one of ordinary skill in the art, however, that these specific details need not be used to practice the present invention. In other instances, well-known structures, interfaces and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

FIG. **1** illustrates a typical computer system **100** in which the present invention operates. The preferred embodiment of

the present invention is implemented on an IBM™ Personal Computer manufactured by IBM Corporation of Armonk, N.Y. An alternate embodiment may be implemented on an RS/6000™ Workstation manufactured by IBM Corporation of Armonk, N.Y. It will be apparent to those of ordinary skill in the art that other computer system architectures may also be employed.

In general, such computer systems as illustrated by FIG. 1 comprise a bus **101** for communicating information, a processor **102** coupled with the bus **101** for processing information, main memory **103** coupled with the bus **101** for storing information and instructions for the processor **102**, a read-only memory **104** coupled with the bus **101** for storing static information and instructions for the processor **102**, a display device **105** coupled with the bus **101** for displaying information for a computer user, an input device **106** coupled with the bus **101** for communicating information and command selections to the processor **102**, and a mass storage device **107**, such as a magnetic disk and associated disk drive, coupled with the bus **101** for storing information and instructions. A data storage medium **108** containing digital information is configured to operate with mass storage device **107** to allow processor **102** access to the digital information on data storage medium **108** via bus **101**.

Processor **102** may be any of a wide variety of general purpose processors or microprocessors such as the Pentium™ microprocessor manufactured by Intel™ Corporation or the RS/6000™ processor manufactured by IBM Corporation. It will be apparent to those of ordinary skill in the art, however, that other varieties of processors may also be used in a particular computer system. Display device **105** may be a liquid crystal device, cathode ray tube (CRT), or other suitable display device. Mass storage device **107** may be a conventional hard disk drive, floppy disk drive, CD-ROM drive, or other magnetic or optical data storage device for reading and writing information stored on a hard disk, a floppy disk, a CD-ROM a magnetic tape, or other magnetic or optical data storage medium. Data storage medium **108** may be a hard disk, a floppy disk, a CD-ROM, a magnetic tape, or other magnetic or optical data storage medium.

In general, processor **102** retrieves processing instructions and data from a data storage medium **108** using mass storage device **107** and downloads this information into random access memory **103** for execution. Processor **102**, then executes an instruction stream from random access memory **103** or read-only memory **104**. Command selections and information input at input device **106** are used to direct the flow of instructions executed by processor **102**. Equivalent input device **106** may also be a pointing device such as a conventional mouse or trackball device. The results of this processing execution are then displayed on display device **105**.

The preferred embodiment of the present invention is implemented as a software module, which may be executed on a computer system such as computer system **100** in a conventional manner. Using well known techniques, the application software of the preferred embodiment is stored on data storage medium **108** and subsequently loaded into and executed within computer system **100**. Once initiated, the software of the preferred embodiment operates in the manner described below.

FIG. **2** illustrates a typical prior art Web server environment. Web client **200** can make URL requests to Web server **201** or Web server **202**. Web servers **201** and **202** include Web server executables, **201**(E) and **202**(E) respectively, that perform the processing of Web client requests. Each Web server may have a number of Web pages **201**(1)–(n) and **202**(1)–(n). Depending on the URL specified by the Web client **200**, the request may be routed by either Web server executable **201**(E) to Web page **201** (1), for example, or from Web server executable **202**(E) to Web page **202** (1). Web client **200** can continue making URL requests to retrieve other Web pages. Web client **200** can also use hyperlinks within each Web page to "jump" to other Web pages or to other locations within the same Web page.

FIG. **3** illustrates this prior art Web server environment in the form of a flow diagram. In processing block **300**, the Web client makes a URL request. This URL request is examined by the Web browser to determine the appropriate Web server to route the request to in processing block **302**. In processing block **304** the request is then transmitted from the Web browser to the appropriate Web server, and in processing block **306** the Web server executable examines the URL to determine whether it is a HTML document or a CGI application. If the request is for an HTML document **308**, then the Web server executable locates the document in processing block **310**. The document is then transmitted back through the requesting Web browser for formatting and display in processing block **312**.

If the URL request is for a CGI application **314**, however, the Web server executable locates the CGI application in processing block **316**. The CGI application then executes and outputs HTML output in processing block **318** and finally, the HTML output is transmitted back to requesting Web browser for formatting and display in processing block **320**.

This prior art Web server environment does not, however, provide any mechanism for managing the Web requests or the Web sites. As Web sites grow, and as the number of Web clients and requests increase, Web site management becomes a crucial need.

For example, a large Web site may receive thousands of requests or "hits" in a single day. Current Web servers process each of these requests on a single machine, namely the Web server machine. Although these machines may be running "multi-threaded" operating systems that allow transactions to be processed by independent "threads," all the threads are nevertheless on a single machine, sharing a processor. As such, the Web executable thread may hand off a request to a processing thread, but both threads will still have to be handled by the processor on the Web server machine. When numerous requests are being simultaneously processed by multiple threads on a single machine, the Web server can slow down significantly and become highly inefficient. The claimed invention addresses this need by utilizing a partitioned architecture to facilitate the creation and management of custom Web sites and servers.

FIG. **4** illustrates one embodiment of the presently claimed invention. Web client **200** issues a URL request that is processed to determined proper routing. In this embodiment, the request is routed to Web server **201**. Instead of Web server executable **201**(E) processing the URL request, however, Interceptor **400** intercepts the request and routes it to Dispatcher **402**. In one embodiment, Interceptor **400** resides on the Web server machine as an extension to Web server **201**. This embodiment is appropriate for Web servers such as Netsite™ from Netscape, that support such extensions. A number of public domain Web servers, such as NCSA™ from the National Center for Supercomputing Applications at the University of Illinois, Urbana-Champaign, however, do not provide support for

this type of extension. Thus, in an alternate embodiment, Interceptor **400** is an independent module, connected via an "intermediate program" to Web server **201**. This intermediate program can be a simple CGI application program that connects Interceptor **400** to Web server **201**. Alternate intermediate programs the perform the same functionality can also be implemented.

In one embodiment of the invention, Dispatcher **402** resides on a different machine than Web server **201**. This embodiment overcomes the limitation described above, in prior art Web servers, wherein all processing is performed by the processor on a single machine. By routing the request to Dispatcher **402** residing on a different machine than the Web server executable **201**(E), the request can then be processed by a different processor than the Web server executable **201**(E). Web server executable **201**(E) is thus free to continue servicing client requests on Web server **201** while the request is processed "off-line," at the machine on which Dispatcher **402** resides.

Dispatcher **402** can, however, also reside on the same machine as the Web server. The Web site administrator has the option of configuring Dispatcher **402** on the same machine as Web server **201**, taking into account a variety of factors pertinent to a particular Web site, such as the size of the Web site, the number of Web pages and the number of hits at the Web site. Although this embodiment will not enjoy the advantage described above, namely off-loading the processing of Web requests from the Web server machine, the embodiment does allow flexibility for a small Web site to grow. For example, a small Web site administrator can use a single machine for both Dispatcher **402** and Web server **201** initially, then off-load Dispatcher **402** onto a separate machine as the Web site grows. The Web site can thus take advantage of other features of the present invention regardless of whether the site has separate machines configured as Web servers and dispatchers.

Dispatcher **402** receives the intercepted request and then dispatches the request to one of a number of Page servers **404** (1)–(n). For example, if Page server **404** (1) receives the dispatched request, it processes the request and retrieves the data from an appropriate data source, such as data source **406**, data source **408**, or data source **410**. Data sources, as used in the present application, include databases, spreadsheets, files and any other type of data repository. Page server **404** (1) can retrieve data from more than one data source and incorporate the data from these multiple data sources in a single Web page.

In one embodiment, each Page server **404**(1)–(n) resides on a separate machine on the network to distribute the processing of the request. Dispatcher **402** maintains a variety of information regarding each Page server on the network, and dispatches requests based on this information. For example, Dispatcher **402** retains dynamic information regarding the data sources that any given Page server can access. Dispatcher **402** thus examines a particular request and determines which Page servers can service the URL request. Dispatcher **402** then hands off the request to the appropriate Page server.

For example, if the URL request requires financial data from data source **408**, dispatcher **402** will first examine an information list. Dispatcher **402** may determine that Page server **404**(3), for example, has access to the requisite data in data source **408**. Dispatcher **402** will thus route the URL request to Page server **404**(3). This "connection caching" functionality is described in more detail below, under the heading "Performance."

Alternately, Dispatcher **402** also has the ability to determine whether a particular Page server already has the necessary data cached in the Page server's page cache (described in more detail below, under the heading "Performance"). Dispatcher **402** may thus determine that Page server **404**(1) and **404**(2) are both logged into Data source **408**, but that Page server **404**(2) has the financial information already cached in Page server **404**(2)'s page cache. In this case, Dispatcher **402** will route the URL request to Page server **404**(2) to more efficiently process the request.

Finally, Dispatcher **402** may determine that a number or all Page servers **404**(1)–(n) are logged into Data source **408**. In this scenario, Dispatcher **402** can examine the number of requests that each Page server is servicing and route the request to the least busy page server. This "load balancing" capability can significantly increase performance at a busy Web site and is discussed in more detail below, under the heading "Scalability".

If, for example, Page server **404**(2), receives the request, Page server **404**(2) will process the request. While Page server **404**(2) is processing the request, Web server executable **201**(E) can concurrently process other Web client requests. This partitioned architecture thus allows both Page server **404**(2) and Web server executable **201**(E) to simultaneously process different requests, thus increasing the efficiency of the Web site. Page server **404**(2) dynamically generates a Web page in response to the Web client request, and the dynamic Web page is then either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

One embodiment of the claimed invention also provides a Web page designer with HTML extensions, or "dyna" tags. These dyna tags provide customized HTML functionality to a Web page designer, to allow the designer to build customized HTML templates that specify the source and placement of retrieved data. For example, in one embodiment, a "dynatext" HTML extension tag specifies a data source and a column name to allow the HTML template to identify the data source to log into and the column name from which to retrieve data. Alternatively, "dyna-anchor" tags allow the designer to build hyperlink queries while "dynablock" tags provide the designer with the ability to iterate through blocks of data. Page servers use these HTML templates to create dynamic Web pages. Then, as described above, these dynamic Web pages are either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

The presently claimed invention provides numerous advantages over prior art Web servers, including advantages in the areas of performance, security, extensibility and scalability.

Performance

One embodiment of the claimed invention utilizes connection caching and page caching to improve performance. Each Page server can be configured to maintain a cache of connections to numerous data sources. For example, as illustrated in FIG. **4**, Page server **404**(1) can retrieve data from data source **406**, data source **408** or data source **410**. Page server **404**(1) can maintain connection cache **412**(1), containing connections to each of data source **406**, data source **408** and data source **410**, thus eliminating connect times from the Page servers to those data sources.

Additionally, another embodiment of the present invention supports the caching of finished Web pages, to optimize

the performance of the data source being utilized. This "page caching" feature, illustrated in FIG. **4** as Page cache **414**, allows the Web site administrator to optimize the performance of data sources by caching Web pages that are repeatedly accessed. Once the Web page is cached, subsequent requests or "hits" will utilize the cached Web page rather than re-accessing the data source. This can radically improve the performance of the data source.

Security

The present invention allows the Web site administrator to utilize multiple levels of security to manage the Web site. In one embodiment, the Page server can utilize all standard encryption and site security features provided by the Web server. In another embodiment, the Page server can be configured to bypass connection caches **412**(1)–(n), described above, for a particular data source and to require entry of a user-supplied identification and password for the particular data source the user is trying to access.

Additionally, another embodiment of the presently claimed invention requires no real-time access of data sources. The Web page caching ability, described above, enables additional security for those sites that want to publish non-interactive content from internal information systems, but do not want real-time Internet accessibility to those internal information systems. In this instance, the Page server can act as a "replication and staging gent" and create Web pages in batches, rather than in real-time. These "replicated" Web pages are then "staged" for access at a later time, and access o the Web pages in this scenario is possible even if the Page server and dispatcher are not present later.

In yet another embodiment, the Page server can make a single pass through a Web library, and compile a Web site that exists in the traditional form of separately available files. A Web library is a collection of related Web books and Web pages. More specifically, the Web library is a hierarchical organization of Web document templates, together with all the associated data source information. Information about an entire Web site is thus contained in a single physical file, thus simplifying the problem of deploying Web sites across multiple Page servers. The process of deploying the Web site in this embodiment is essentially a simple copy of a single file.

Extensibility

One embodiment of the present invention provides the Web site administrator with Object Linking and Embedding (OLE) 2.0 extensions to extend the page creation process. These OLE 2.0 extensions also allow information submitted over the Web to be processed with user-supplied functionality. Utilizing development tools such as Visual Basic, Visual C++ or PowerBuilder that support the creation of OLE 2.0 automation, the Web site administrator can add features and modify the behavior of the Page servers described above. This extensibility allows one embodiment of the claimed invention to be incorporated with existing technology to develop an infinite number of custom web servers.

For example, OLE 2.0 extensions allow a Web site administrator to encapsulate existing business rules in an OLE 2.0 automation interface, to be accessed over the Web. One example of a business rule is the steps involved in the payoff on an installment or mortgage loan. The payoff may involve, for example, taking into account the current balance, the date and the interest accrued since the last payment. Most organizations already have this type of business rule implemented using various applications, such as Visual Basic for client-server environments, or CICS programs on mainframes. If these applications are OLE 2.0 compliant, the Page server "dynaobject" HTML extension tag can be used to encapsulated the application in an OLE 2.0 automation interface. The Page server is thus extensible, and can incorporate the existing application with the new Page server functionality.

Scalability

One embodiment of the claimed invention allows "plug and play" scalability. As described above, referring to FIG. **4**, Dispatcher **402** maintains information about all the Page servers configured to be serviced by Dispatcher **402**. Any number of Page servers can thus be "plugged" into the configuration illustrated in FIG. **4**, and the Page servers will be instantly activated as the information is dynamically updated in Dispatcher **402**. The Web site administrator can thus manage the overhead of each Page server and modify each Page server's load, as necessary, to improve performance. In this manner, each Page server will cooperate with other Page servers within a multi-server environment. Dispatcher **402** can examine the load on each Page server and route new requests according to each Page server's available resources. This "load-balancing" across multiple Page servers can significantly increase a Web site's performance.

FIG. **5** illustrates the processing of a Web browser request in the form of a flow diagram, according to one embodiment of the presently claimed invention. A Web browser sends a URL request to a Web server in processing block **500**. In processing block **502**, the Web server receives the URL request, and an interceptor then intercepts the handling of the request in processing block **504**. The interceptor connects to a dispatcher and sends the URL request to the dispatcher in processing block **506**. In processing block **508**, the dispatcher determines which Page servers can handle the request. The dispatcher also determines which Page server is processing the fewest requests in processing block **510**, and in processing block **512**, the dispatcher sends the URL request to an appropriate Page server. The Page server receives the request and produces an HTML document in processing block **514**. The Page server then responds to the dispatcher with notification of the name of the cached HTML document in processing block **516**. In processing block **518**, the dispatcher responds to the interceptor with the document name, and the interceptor then replaces the requested URL with the newly generated HTML document in processing block **520**. The Web server then sends the new HTML document to the requesting client in processing block **522**. Finally, the Web browser receives and displays the HTML document created by the Page server at processing block **524**.

Thus, a method and apparatus for creating and managing custom Web sites is disclosed. These specific arrangements and methods described herein are merely illustrative of the principles of the present invention. Numerous modifications in form and detail may be made by those of ordinary skill in the art without departing from the scope of the present invention. Although this invention has been shown in relation to a particular preferred embodiment, it should not be considered so limited. Rather, the present invention is limited only by the scope of the appended claims.

We claim:

1. A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of:

routing said request from said Web server to a page server, said page server receiving said request and releasing

said Web server to process other requests, wherein said routing step further includes the steps of intercepting said request at said Web server, routing said request from said Web server to a dispatcher, and dispatching said request to said page server;

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from one or more data sources.

2. The computer-implemented method in claim **1** wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

3. The computer-implemented method in claim **2** wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources.

4. The computer-implemented method in claim **3** wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

5. The computer-implemented method in claim **3** wherein said step of processing said request includes the step of logging into said one or more data sources.

6. The computer-implemented method in claim **3** wherein said step of dynamically generating said Web page includes the step of maintaining a page cache containing said Web page.

7. The computer-implemented method in claim **3** wherein said page server includes custom HTML extension templates for configuring said Web page.

8. The computer-implemented method in claim **7** wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said custom HTML extension templates.

9. A networked system for managing a dynamic Web page generation request, said system comprising:

one or more data sources;

a page server having a processing means;

a first computer system including means for generating said request; and

a second computer system including means for receiving said request from said first computer, said second computer system also including a router, said router routing said request from said second computer system to said page server, wherein said routing further includes intercepting said request at said second computer, routing said request from said second computer to a dispatcher, and dispatching said request to said page server said page server receiving said request and releasing said second computer system to process other requests, said page server processing means processing said request and dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from said one or more data sources.

10. The networked system in claim **9** wherein said router in said second computer system includes:

an interceptor intercepting said request at said second computer system and routing said request; and

a dispatcher receiving said routed request from said interceptor and dispatching said request to said page server.

11. A machine readable medium having stored thereon data representing sequences of instructions, which when executed by a computer system, cause said computer system to perform the steps of:

routing a dynamic Web page generation request from a Web server to a page server, said page server receiving said request and releasing said Web server to process other requests wherein said routing step further includes the steps of intercepting said request at said Web server, routing said request from said Web server to a dispatcher, and dispatching said request to said page server;

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

dynamically generating a Web page, said Web page including data retrieved from one or more data sources.

* * * * *

Exhibit C

US006415335B1

(12) **United States Patent**
Lowery et al.

(10) **Patent No.: US 6,415,335 B1**
(45) **Date of Patent: *Jul. 2, 2002**

(54) **SYSTEM AND METHOD FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS**

(75) Inventors: **Keith Lowery**, Richardson; **Andrew B. Levine**, Plano; **Ronald L. Howell**, Rowlett, all of TX (US)

(73) Assignee: **epicRealm Operating Inc.**, Richardson, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/234,048**

(22) Filed: **Jan. 19, 1999**

Related U.S. Application Data

(62) Division of application No. 08/636,477, filed on Apr. 23, 1996, now Pat. No. 5,894,554.

(51) **Int. Cl.**[7] **G06F 13/14**; G06F 13/20

(52) **U.S. Cl.** **710/5**; 710/7; 709/219; 709/223; 709/238

(58) **Field of Search** 709/238, 223, 709/219; 710/5, 7, 20–21

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,866,706 A		9/1989	Christophersen et al.	370/85.7
5,341,499 A		8/1994	Doragh	395/700
5,392,400 A		2/1995	Berkowitz et al.	395/200
5,404,522 A		4/1995	Carmon et al.	395/650
5,404,523 A		4/1995	DellaFera et al.	395/650
5,404,527 A	*	4/1995	Irwin et al.	395/700
5,452,460 A		9/1995	Distelberg et al.	395/700
5,532,838 A		7/1996	Barbari	358/400
5,701,463 A	*	12/1997	Malcolm	395/610
5,751,956 A		5/1998	Kirsch	395/200.33
5,752,246 A	*	5/1998	Rogers et al.	707/10
5,754,772 A	*	5/1998	Leaf	395/200.33
5,761,673 A	*	6/1998	Bookman et al.	707/104
5,774,660 A		6/1998	Brendel et al.	395/200.31
5,774,668 A		6/1998	Choquier et al.	395/200.53

OTHER PUBLICATIONS

Hoffner, 'Inter-operability and distributed application platform design', Web URL:http://www.ansa.co.uk/, 1995, pp. 342–356.*

Mourad et al, 'Scalable Web Server Architectures', IEEE, Jun. 1997, pp. 12–16.*

Dias et al, 'A Scalable and Highly Available Web Server', IEEE, 1996, pp. 85–92.*

'Single System Image and Load Balancing for Network Access to a Loosely Coupled Complex', IBM TDB, vol. 34, Feb. 1992, pp. 464–467.*

Dias, Daniel M., et al.; A Scalable and Highly Available Web Server; IBM Research Division; T.J. Watson Research Center; 7 pages.

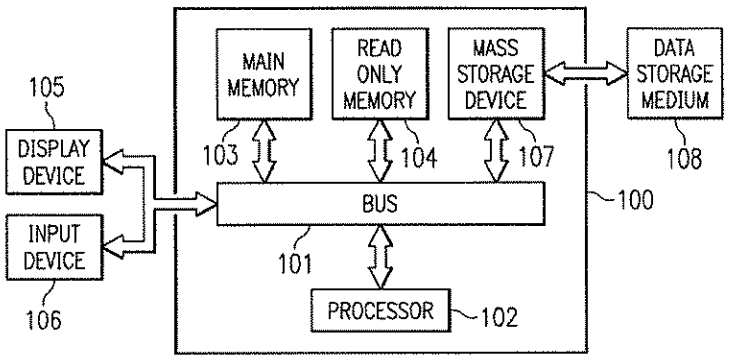
(List continued on next page.)

Primary Examiner—Jeffrey Gaffin
Assistant Examiner—Rehana Perveen
(74) *Attorney, Agent, or Firm*—Baker Botts L.L.P.

(57) **ABSTRACT**

The present invention teaches a method and apparatus for creating and managing custom Web sites. Specifically, one embodiment of the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources.

29 Claims, 4 Drawing Sheets



OTHER PUBLICATIONS

Andresen, Daniel, Et Al.; Scalability Issues for High Performance Digital Libraries on the World Wide Web; Department of Computer Science; University of California at Santa Barbara; 10 pages.

Andresen, Daniel, Et Al.; SWEB: Towards a Scalable World Wide Web Server on Multicomputers; Department of Computer Science; University of California at Santa Barbara; 7 pages.

Holmedahl, Vegard; Et Al.; Cooperative Caching of Dynamic Content on a Distributed Web Server; Department of Computer Science; University of California at Santa Barbara; 8 pages.

Overson, Nicole; NeXT Ships WebObjects—On Time—As Promised; Deja.com: NeXT Ships WebObjects—On Time—As Promishttp://X28..deja.com/=dnc/ST_m=ps...EXT=927585438. 1744765032&hitnum=33.

International Search Report; 7 pages; dated Aug. 21, 1997.

Birman, Kenneth P. and van Renesse, Robbert; Software for Reliable Networks; Scientific American; May 1996; pp. 64–69.

"Beyond the Web: Excavating the Real World Via Mosaic"; Goldberg et al.; Second International WWW Conference; Oct. 17, 1994.

* cited by examiner

FIG. 1

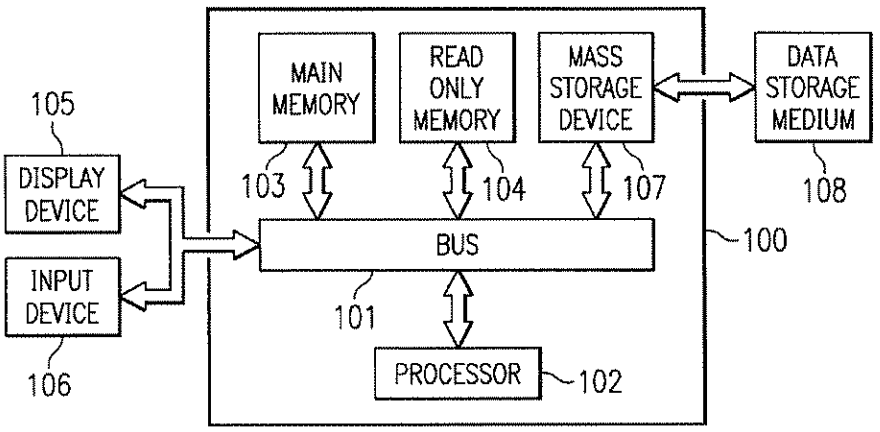


FIG. 2
(PRIOR ART)

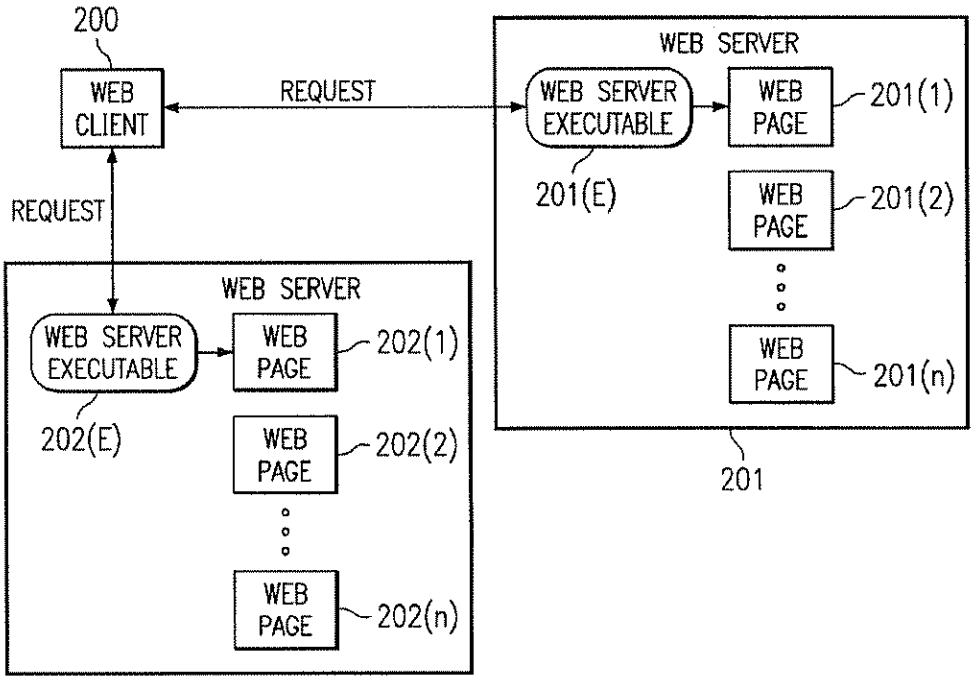
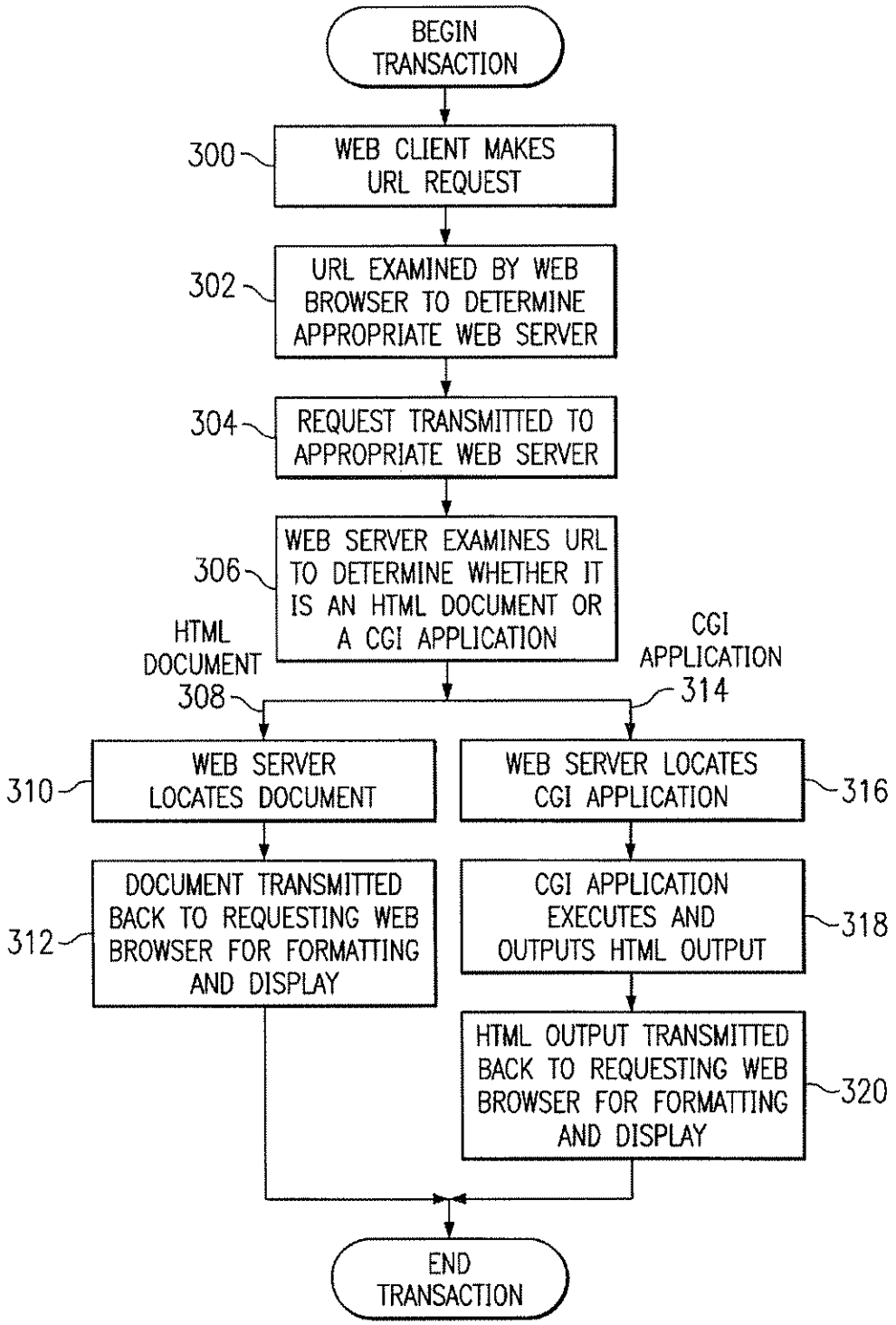


FIG. 3
(PRIOR ART)



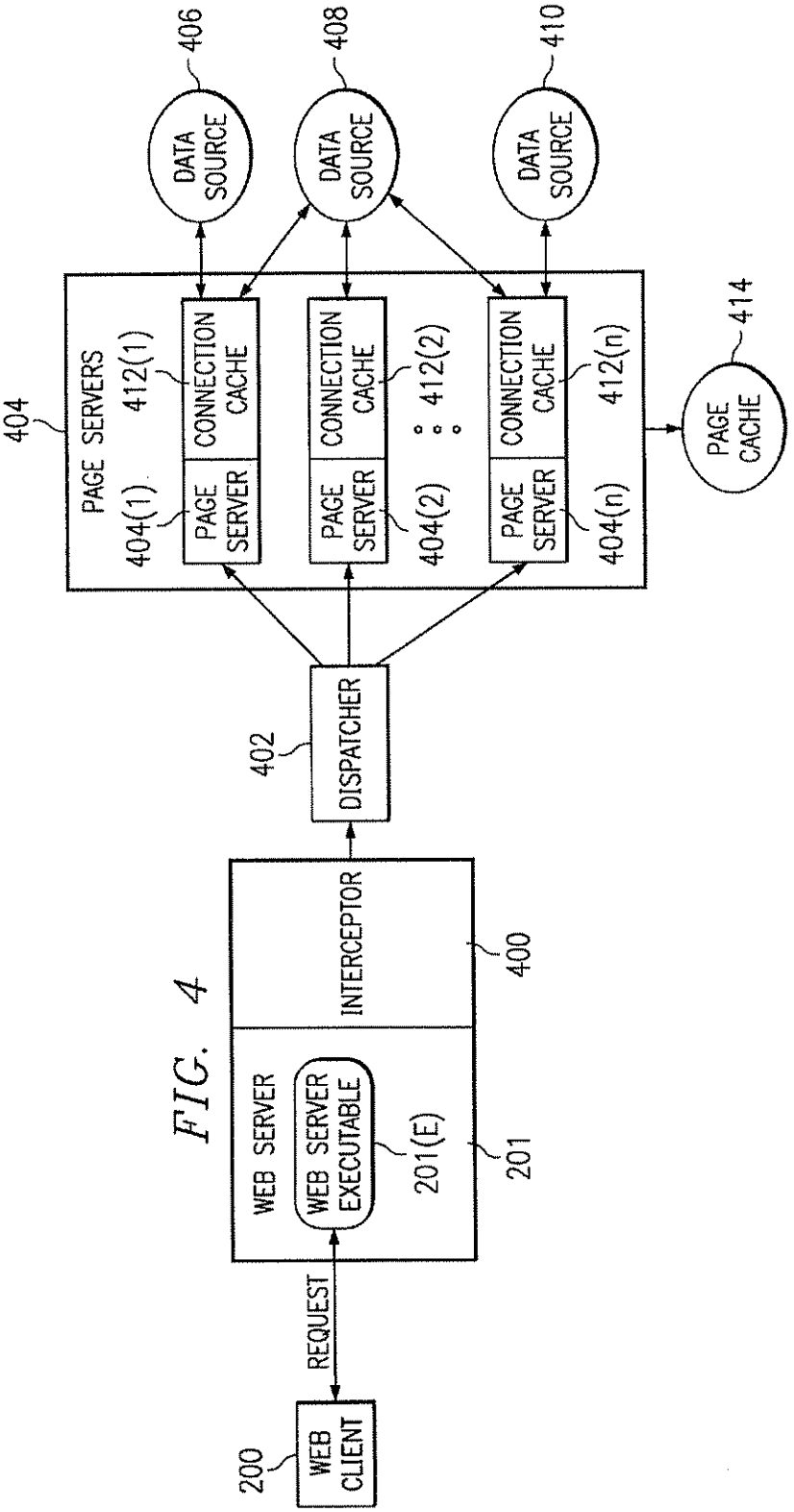
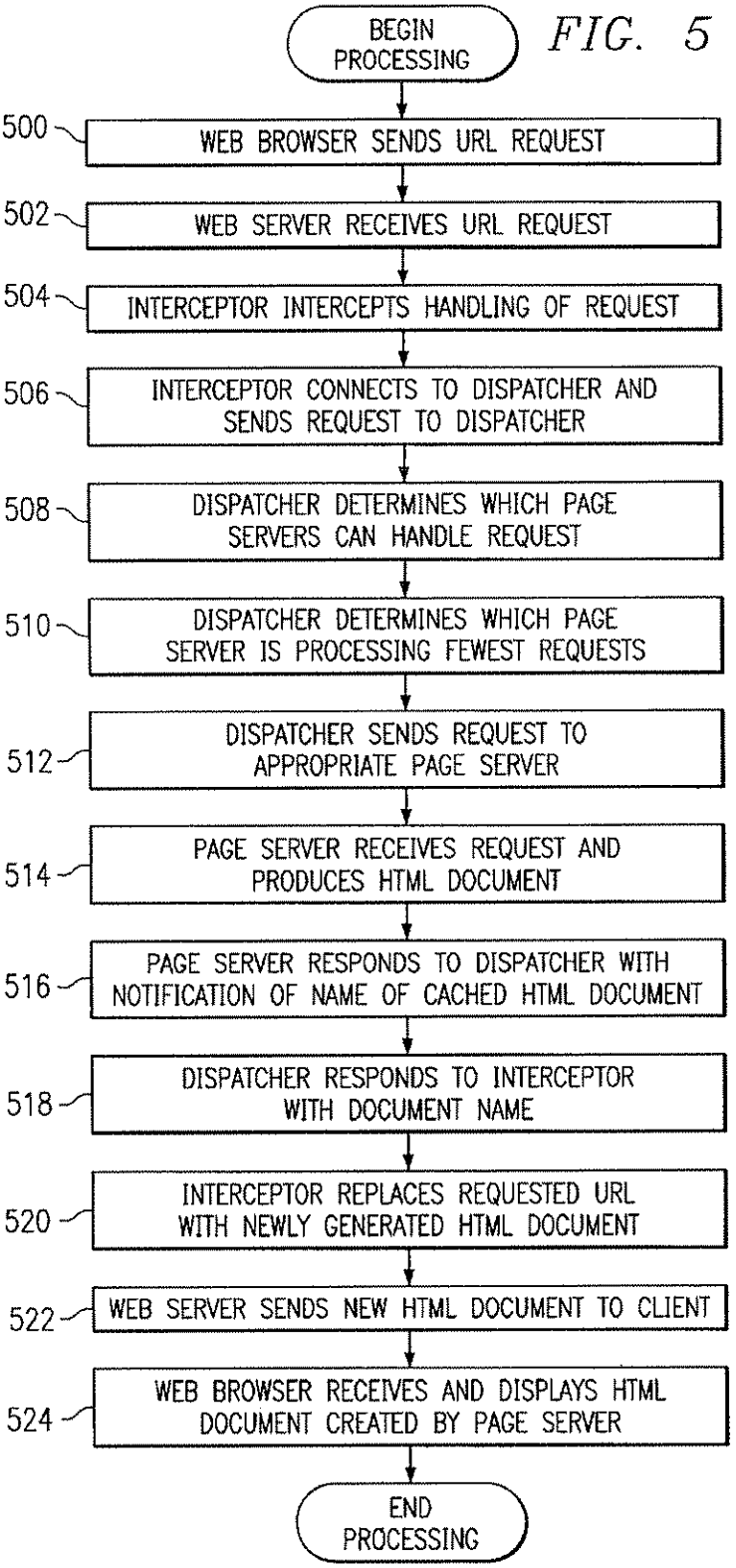

406
408
410
DATA SOURCE
DATA SOURCE
DATA SOURCE
404
PAGE SERVERS
412(1)
CONNECTION CACHE
PAGE SERVER
404(1)
CONNECTION CACHE
PAGE SERVER
404(2)
412(2)
CONNECTION CACHE
PAGE SERVER
404(n)
412(n)
414
PAGE CACHE
402
DISPATCHER
INTERCEPTOR
400
WEB SERVER
WEB SERVER EXECUTABLE
201(E)
201
REQUEST
200
WEB CLIENT

FIG. 4

FIG. 5


BEGIN PROCESSING
500
WEB BROWSER SENDS URL REQUEST
502
WEB SERVER RECEIVES URL REQUEST
504
INTERCEPTOR INTERCEPTS HANDLING OF REQUEST
506
INTERCEPTOR CONNECTS TO DISPATCHER AND SENDS REQUEST TO DISPATCHER
508
DISPATCHER DETERMINES WHICH PAGE SERVERS CAN HANDLE REQUEST
510
DISPATCHER DETERMINES WHICH PAGE SERVER IS PROCESSING FEWEST REQUESTS
512
DISPATCHER SENDS REQUEST TO APPROPRIATE PAGE SERVER
514
PAGE SERVER RECEIVES REQUEST AND PRODUCES HTML DOCUMENT
516
PAGE SERVER RESPONDS TO DISPATCHER WITH NOTIFICATION OF NAME OF CACHED HTML DOCUMENT
518
DISPATCHER RESPONDS TO INTERCEPTOR WITH DOCUMENT NAME
520
INTERCEPTOR REPLACES REQUESTED URL WITH NEWLY GENERATED HTML DOCUMENT
522
WEB SERVER SENDS NEW HTML DOCUMENT TO CLIENT
524
WEB BROWSER RECEIVES AND DISPLAYS HTML DOCUMENT CREATED BY PAGE SERVER
END PROCESSING

SYSTEM AND METHOD FOR MANAGING DYNAMIC WEB PAGE GENERATION REQUESTS

This application is a division of Ser. No. 08/636,477, filed Apr. 23, 1996, now U.S. Pat. No. 5,894,554.

FIELD OF THE INVENTION

The present invention relates to the field of Internet technology. Specifically, the present invention relates to the creation and management of custom World Wide Web sites.

DESCRIPTION OF RELATED ART

The World Wide Web (the Web) represents all of the computers on the Internet that offer users access to information on the Internet via interactive documents or Web pages. These Web pages contain hypertext links that are used to connect any combination of graphics, audio, video and text, in a non-linear, non-sequential manner. Hypertext links are created using a special software language known as HyperText Mark-Up Language (HTML).

Once created, Web pages reside on the Web, on Web servers or Web sites. A Web site can contain numerous Web pages. Web client machines running Web browsers can access these Web pages at Web sites via a communications protocol known as HyperText Transport Protocol (HTTP). Web browsers are software interfaces that run on World Wide Web clients to allow access to Web sites via a simple user interface. A Web browser allows a Web client to request a particular Web page from a Web site by specifying a Uniform Resource Locator (URL). A URL is a Web address that identifies the Web page and its location on the Web. When the appropriate Web site receives the URL, the Web page corresponding to the requested URL is located, and if required, HTML output is generated. The HTML output is then sent via HTTP to the client for formatting on the client's screen.

Although Web pages and Web sites are extremely simple to create, the proliferation of Web sites on the Internet highlighted a number of problems. The scope and ability of a Web page designer to change the content of the Web page was limited by the static nature of Web pages. Once created, a Web page remained static until it was manually modified. This in turn limited the ability of Web site managers to effectively manage their Web sites.

The Common Gateway Interface (CGI) standard was developed to resolve the problem of allowing dynamic content to be included in Web pages. CGI "calls" or procedures enable applications to generate dynamically created HTML output, thus creating Web pages with dynamic content. Once created, these CGI applications do not have to be modified in order to retrieve "new" or dynamic data. Instead, when the Web page is invoked, CGI "calls" or procedures are used to dynamically retrieve the necessary data and to generate a Web page.

CGI applications also enhanced the ability of Web site administrators to manage Web sites. Administrators no longer have to constantly update static Web pages. A number of vendors have developed tools for CGI based development, to address the issue of dynamic Web page generation. Companies like Spider™ and Bluestone™, for example, have each created development tools for CGI-based Web page development. Another company, Haht Software™, has developed a Web page generation tool that uses a BASIC-like scripting language, instead of a CGI scripting language.

Tools that generate CGI applications do not, however, resolve the problem of managing numerous Web pages and requests at a Web site. For example, a single company may maintain hundreds of Web pages at their Web site. Current Web server architecture also does not allow the Web server to efficiently manage the Web page and process Web client requests. Managing these hundreds of Web pages in a coherent manner and processing all requests for access to the Web pages is thus a difficult task. Existing development tools are limited in their capabilities to facilitate dynamic Web page generation, and do not address the issue of managing Web requests or Web sites.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide a method and apparatus for creating and managing custom Web sites. Specifically, the present invention claims a method and apparatus for managing dynamic web page generation requests.

In one embodiment, the present invention claims a computer-implemented method for managing a dynamic Web page generation request to a Web server, the computer-implemented method comprising the steps of routing the request from the Web server to a page server, the page server receiving the request and releasing the Web server to process other requests, processing the request, the processing being performed by the page server concurrently with the Web server, as the Web server processes the other requests, and dynamically generating a Web page in response to the request, the Web page including data dynamically retrieved from one or more data sources. Other embodiments also include connection caches to the one or more data sources, page caches for each page server, and custom HTML extension templates for configuring the Web page.

Other objects, features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a typical computer system in which the present invention operates.

FIG. **2** illustrates a typical prior art Web server environment.

FIG. **3** illustrates a typical prior art Web server environment in the form of a flow diagram.

FIG. **4** illustrates one embodiment of the presently claimed invention.

FIG. **5** illustrates the processing of a Web browser request in the form of a flow diagram, according to one embodiment of the presently claimed invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention relates to a method and apparatus for creating and managing custom Web sites. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent to one of ordinary skill in the art, however, that these specific details need not be used to practice the present invention. In other instances, well-known structures, interfaces and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

FIG. **1** illustrates a typical computer system **100** in which the present invention operates. The preferred embodiment of

the present invention is implemented on an IBM™ Personal Computer manufactured by IBM Corporation of Armonk, New York. An alternate embodiment may be implemented on an RS/6000™ Workstation manufactured by IBM Corporation of Armonk, New York. It will be apparent to those of ordinary skill in the art that other computer system architectures may also be employed.

In general, such computer systems as illustrated by FIG. **1** comprise a bus **101** for communicating information, a processor **102** coupled with the bus **101** for processing information, main memory **103** coupled with the bus **101** for storing information and instructions for the processor **102**, a read-only memory **104** coupled with the bus **101** for storing static information and instructions for the processor **102**, a display device **105** coupled with the bus **101** for displaying information for a computer user, an input device **106** coupled with the bus **101** for communicating information and command selections to the processor **102**, and a mass storage device **107**, such as a magnetic disk and associated disk drive, coupled with the bus **101** for storing information and instructions. A data storage medium **108** containing digital information is configured to operate with mass storage device **107** to allow processor **102** access to the digital information on data storage medium **108** via bus **101**.

Processor **102** may be any of a wide variety of general purpose processors or microprocessors such as the Pentium™ microprocessor manufactured by Intel™ Corporation or the RS/6000™ processor manufactured by IBM Corporation. It will be apparent to those of ordinary skill in the art, however, that other varieties of processors may also be used in a particular computer system. Display device **105** may be a liquid crystal device, cathode ray tube (CRT), or other suitable display device. Mass storage device **107** may be a conventional hard disk drive, floppy disk drive, CD-ROM drive, or other magnetic or optical data storage device for reading and writing information stored on a hard disk, a floppy disk, a CD-ROM a magnetic tape, or other magnetic or optical data storage medium. Data storage medium **108** may be a hard disk, a floppy disk, a CD-ROM, a magnetic tape, or other magnetic or optical data storage medium.

In general, processor **102** retrieves processing instructions and data from a data storage medium **108** using mass storage device **107** and downloads this information into random access memory **103** for execution. Processor **102**, then executes an instruction stream from random access memory **103** or read-only memory **104**. Command selections and information input at input device **106** are used to direct the flow of instructions executed by processor **102**. Equivalent input device **106** may also be a pointing device such as a conventional mouse or trackball device. The results of this processing execution are then displayed on display device **105**.

The preferred embodiment of the present invention is implemented as a software module, which may be executed on a computer system such as computer system **100** in a conventional manner. Using well known techniques, the application software of the preferred embodiment is stored on data storage medium **108** and subsequently loaded into and executed within computer system **100**. Once initiated, the software of the preferred embodiment operates in the manner described below.

FIG. **2** illustrates a typical prior art Web server environment. Web client **200** can make URL requests to Web server **201** or Web server **202**. Web servers **201** and **202** include Web server executables, **201**(E) and **202**(E) respectively, that perform the processing of Web client requests. Each Web server may have a number of Web pages **201**(1)–(n) and **202**(1)–(n). Depending on the URL specified by the Web client **200**, the request may be routed by either Web server executable **201**(E) to Web page **201** (1), for example, or from Web server executable **202**(E) to Web page **202** (1). Web client **200** can continue making URL requests to retrieve other Web pages. Web client **200** can also use hyperlinks within each Web page to "jump" to other Web pages or to other locations within the same Web page.

FIG. **3** illustrates this prior art Web server environment in the form of a flow diagram. In processing block **300**, the Web client makes a URL request. This URL request is examined by the Web browser to determine the appropriate Web server to route the request to in processing block **302**. In processing block **304** the request is then transmitted from the Web browser to the appropriate Web server, and in processing block **306** the Web server executable examines the URL to determine whether it is a HTML document or a CGI application. If the request is for an HTML document **308**, then the Web server executable locates the document in processing block **310**. The document is then transmitted back through the requesting Web browser for formatting and display in processing block **312**.

If the URL request is for a CGI application **314**, however, the Web server executable locates the CGI application in processing block **316**. The CGI application then executes and outputs HTML output in processing block **318** and finally, the HTML output is transmitted back to requesting Web browser for formatting and display in processing block **320**.

This prior art Web server environment does not, however, provide any mechanism for managing the Web requests or the Web sites. As Web sites grow, and as the number of Web clients and requests increase, Web site management becomes a crucial need.

For example, a large Web site may receive thousands of requests or "hits" in a single day. Current Web servers process each of these requests on a single machine, namely the Web server machine. Although these machines may be running "multi-threaded" operating systems that allow transactions to be processed by independent "threads," all the threads are nevertheless on a single machine, sharing a processor. As such, the Web executable thread may hand off a request to a processing thread, but both threads will still have to be handled by the processor on the Web server machine. When numerous requests are being simultaneously processed by multiple threads on a single machine, the Web server can slow down significantly and become highly inefficient. The claimed invention addresses this need by utilizing a partitioned architecture to facilitate the creation and management of custom Web sites and servers.

FIG. **4** illustrates one embodiment of the presently claimed invention. Web client **200** issues a URL request that is processed to determined proper routing. In this embodiment, the request is routed to Web server **201**. Instead of Web server executable **201**(E) processing the URL request, however, Interceptor **400** intercepts the request and routes it to Dispatcher **402**. In one embodiment, Interceptor **400** resides on the Web server machine as an extension to Web server **201**. This embodiment is appropriate for Web servers such as Netsite™ from Netscape, that support such extensions. A number of public domain Web servers, such as NCSA™ from the National Center for Supercomputing Applications at the University of Illinois, Urbana-Champaign, however, do not provide support for

this type of extension. Thus, in an alternate embodiment, Interceptor **400** is an independent module, connected via an "intermediate program" to Web server **201**. This intermediate program can be a simple CGI application program that connects Interceptor **400** to Web server **201**. Alternate intermediate programs the perform the same functionality can also be implemented.

In one embodiment of the invention, Dispatcher **402** resides on a different machine than Web server **201**. This embodiment overcomes the limitation described above, in prior art Web servers, wherein all processing is performed by the processor on a single machine. By routing the request to Dispatcher **402** residing on a different machine than the Web server executable **201**(E), the request can then be processed by a different processor than the Web server executable **201**(E). Web server executable **201**(E) is thus free to continue servicing client requests on Web server **201** while the request is processed "off-line," at the machine on which Dispatcher **402** resides.

Dispatcher **402** can, however, also reside on the same machine as the Web server. The Web site administrator has the option of configuring Dispatcher **402** on the same machine as Web server **201**, taking into account a variety of factors pertinent to a particular Web site, such as the size of the Web site, the number of Web pages and the number of hits at the Web site. Although this embodiment will not enjoy the advantage described above, namely off-loading the processing of Web requests from the Web server machine, the embodiment does allow flexibility for a small Web site to grow. For example, a small Web site administrator can use a single machine for both Dispatcher **402** and Web server **201** initially, then off-load Dispatcher **402** onto a separate machine as the Web site grows. The Web site can thus take advantage of other features of the present invention regardless of whether the site has separate machines configured as Web servers and dispatchers.

Dispatcher **402** receives the intercepted request and then dispatches the request to one of a number of Page servers **404** (1)–(n). For example, if Page server **404** (1) receives the dispatched request, it processes the request and retrieves the data from an appropriate data source, such as data source **406**, data source **408**, or data source **410**. Data sources, as used in the present application, include databases, spreadsheets, files and any other type of data repository. Page server **404** (1) can retrieve data from more than one data source and incorporate the data from these multiple data sources in a single Web page.

In one embodiment, each Page server **404**(1)–(n) resides on a separate machine on the network to distribute the processing of the request. Dispatcher **402** maintains a variety of information regarding each Page server on the network, and dispatches requests based on this information. For example, Dispatcher **402** retains dynamic information regarding the data sources that any given Page server can access. Dispatcher **402** thus examines a particular request and determines which Page servers can service the URL request. Dispatcher **402** then hands off the request to the appropriate Page server.

For example, if the URL request requires financial data from data source **408**, dispatcher **402** will first examine an information list. Dispatcher **402** may determine that Page server **404**(3), for example, has access to the requisite data in data source **408**. Dispatcher **402** will thus route the URL request to Page server **404**(3). This "connection caching" functionality is described in more detail below, under the heading "Performance." Alternately, Dispatcher **402** also has the ability to determine whether a particular Page server already has the necessary data cached in the Page server's page cache (described in more detail below, under the heading "Performance"). Dispatcher **402** may thus determine that Page server **404**(1) and **404**(2) are both logged into Data source **408**, but that Page server **404**(2) has the financial information already cached in Page server **404**(2)'s page cache. In this case, Dispatcher **402** will route the URL request to Page server **404**(2) to more efficiently process the request.

Finally, Dispatcher **402** may determine that a number or all Page servers **404**(1)–(n) are logged into Data source **408**. In this scenario, Dispatcher **402** can examine the number of requests that each Page server is servicing and route the request to the least busy page server. This "load balancing" capability can significantly increase performance at a busy Web site and is discussed in more detail below, under the heading "Scalability".

If, for example, Page server **404**(2), receives the request, Page server **404**(2) will process the request. While Page server **404**(2) is processing the request, Web server executable **201**(E) can concurrently process other Web client requests. This partitioned architecture thus allows both Page server **404**(2) and Web server executable **201**(E) to. simultaneously process different requests, thus increasing the efficiency of the Web site. Page server **404**(2) dynamically generates a Web page in response to the Web client request, and the dynamic Web page is then either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

One embodiment of the claimed invention also provides a Web page designer with HTML extensions, or "dyna" tags. These dyna tags provide customized HTML functionality to a Web page designer, to allow the designer to build customized HTML templates that specify the source and placement of retrieved data. For example, in one embodiment, a "dynatext" HTML extension tag specifies a data source and a column name to allow the HTML template to identify the data source to log into and the column name from which to retrieve data. Alternatively, "dyna-anchor" tags allow the designer to build hyperlink queries while "dynablock" tags provide the designer with the ability to iterate through blocks of data. Page servers use these HTML templates to create dynamic Web pages. Then, as described above, these dynamic Web pages are either transmitted back to requesting Web client **200** or stored on a machine that is accessible to Web server **201**, for later retrieval.

The presently claimed invention provides numerous advantages over prior art Web servers, including advantages in the areas of performance, security, extensibility and scalability

Performance

One embodiment of the claimed invention utilizes connection caching and page caching to improve performance. Each Page server can be configured to maintain a cache of connections to numerous data sources. For example, as illustrated in FIG. **4**, Page server **404**(1) can retrieve data from data source **406**, data source **408** or data source **410**. Page server **404**(1) can maintain connection cache **412**(1), containing connections to each of data source **406**, data source **408** and data source **410**, thus eliminating connect times from the Page servers to those data sources.

Additionally, another embodiment of the present invention supports the caching of finished Web pages, to optimize the performance of the data source being utilized. This "page

caching" feature, illustrated in FIG. **4** as Page cache **414**, allows the Web site administrator to optimize the performance of data sources by caching Web pages that are repeatedly accessed. Once the Web page is cached, subsequent requests or "hits" will utilize the cached Web page rather than re-accessing the data source. This can radically improve the performance of the data source.

Security

The present invention allows the Web site administrator to utilize multiple levels of security to manage the Web site. In one embodiment, the Page server can utilize all standard encryption and site security features provided by the Web server. In another embodiment, the Page server can be configured to bypass connection caches **412(1)**–(n), described above, for a particular data source and to require entry of a user-supplied identification and password for the particular data source the user is trying to access.

Additionally, another embodiment of the presently claimed invention requires no real-time access of data sources. The Web page caching ability, described above, enables additional security for those sites that want to publish non-interactive content from internal information systems, but do not want real-time Internet accessibility to those internal information systems. In this instance, the Page server can act as a "replication and staging agent" and create Web pages in batches, rather than in real-time. These "replicated" Web pages are then "staged" for access at a later time, and access to the Web pages in this scenario is possible even if the Page server and dispatcher are not present later.

In yet another embodiment, the Page server can make a single pass through a Web library, and compile a Web site that exists in the traditional form of separately available files. A Web library is a collection of related Web books and Web pages. More specifically, the Web library is a hierarchical organization of Web document templates, together with all the associated data source information. Information about an entire Web site is thus contained in a single physical file, thus simplifying the problem of deploying Web sites across multiple Page servers. The process of deploying the Web site in this embodiment is essentially a simple copy of a single file.

Extensibility

One embodiment of the present invention provides the Web site administrator with Object Linking and Embedding (OLE) 2.0 extensions to extend the page creation process. These OLE 2.0 extensions also allow information submitted over the Web to be processed with user-supplied functionality. Utilizing development tools such as Visual Basic, Visual C++ or PowerBuilder that support the creation of OLE 2.0 automation, the Web site administrator can add features and modify the behavior of the Page servers described above. This extensibility allows one embodiment of the claimed invention to be incorporated with existing technology to develop an infinite number of custom web servers.

For example, OLE 2.0 extensions allow a Web site administrator to encapsulate existing business rules in an OLE 2.0 automation interface, to be accessed over the Web. One example of a business rule is the steps involved in the payoff on an installment or mortgage loan. The payoff may involve, for example, taking into account the current balance, the date and the interest accrued since the last payment. Most organizations already have this type of business rule implemented using various applications, such as Visual Basic for client-server environments, or CICS programs on mainframes. If these applications are OLE 2.0 compliant, the Page server "dynaobject" HTML extension tag can be used to encapsulated the application in an OLE 2.0 automation interface. The Page server is thus extensible, and can incorporate the existing application with the new Page server functionality.

Scalability

One embodiment of the claimed invention allows "plug and play" scalability. As described above, referring to FIG. **4**, Dispatcher **402** maintains information about all the Page servers configured to be serviced by Dispatcher **402**. Any number of Page servers can thus be "plugged" into the configuration illustrated in FIG. **4**, and the Page servers will be instantly activated as the information is dynamically updated in Dispatcher **402**. The Web site administrator can thus manage the overhead of each Page server and modify each Page server's load, as necessary, to improve performance. In this manner, each Page server will cooperate with other Page servers within a multi-server environment. Dispatcher **402** can examine the load on each Page server and route new requests according to each Page server's available resources. This "load-balancing" across multiple Page servers can significantly increase a Web site's performance.

FIG. **5** illustrates the processing of a Web browser request in the form of a flow diagram, according to one embodiment of the presently claimed invention. A Web browser sends a URL request to a Web server in processing block **500**. In processing block **502**, the Web server receives the URL request, and an interceptor then intercepts the handling of the request in processing block **504**. The interceptor connects to a dispatcher and sends the URL request to the dispatcher in processing block **506**. In processing block **508**, the dispatcher determines which Page servers can handle the request. The dispatcher also determines which Page server is processing the fewest requests in processing block **510**, and in processing block **512**, the dispatcher sends the URL request to an appropriate Page server. The Page server receives the request and produces an HTML document in processing block **514**. The Page server then responds to the dispatcher with notification of the name of the cached HTML document in processing block **516**. In processing block **518**, the dispatcher responds to the interceptor with the document name, and the interceptor then replaces the requested URL with the newly generated HTML document in processing block **520**. The Web server then sends the new HTML document to the requesting client in processing block **522**. Finally, the Web browser receives and displays the HTML document created by the Page server at processing block **524**.

Thus, a method and apparatus for creating and managing custom Web sites is disclosed. These specific arrangements and methods described herein are merely illustrative of the principles of the present invention. Numerous modifications in form and detail may be made by those of ordinary skill in the art without departing from the scope of the present invention. Although this invention has been shown in relation to a particular preferred embodiment, it should not be considered so limited. Rather, the present invention is limited only by the scope of the appended claims.

We claim:

1. A computer-implemented method for managing a dynamic Web page generation request to a Web server, said computer-implemented method comprising the steps of:

routing a request from a Web server to a page server, said page server receiving said request and releasing said

Web server to process other requests wherein said routing step further includes the steps of:

intercepting said request at said Web server and routing said request to said page server;

processing said request, said processing being performed by said page server while said Web server concurrently processes said other requests; and

dynamically generating a Web page in response to said request, said Web page including data dynamically retrieved from one or more data sources.

2. The computer-implemented method in claim 1 wherein said step of routing said request includes the steps of:

routing said request from said Web server to a dispatcher; and

dispatching said request to said page server.

3. The computer-implemented method in claim 1 wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

4. The computer-implemented method in claim 1 wherein said step of dynamically generating said Web page includes the step of dynamically retrieving said data from said one or more data sources.

5. The computer-implemented method in claim 1 wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

6. The computer-implemented method in claim 1 wherein said step of processing said request includes the step of logging into said one or more data sources.

7. The computer-implemented method in claim 1 wherein said step of dynamically generating said Web page includes the step of maintaining a page cache containing said Web page.

8. The computer-implemented method in claim 1 wherein said page server includes tag-based text templates for configuring said Web page.

9. The computer-implemented method in claim 8 wherein said step of processing said request further includes the step of inserting said-dynamically retrieved data from said one or more data sources into said tag-based text templates.

10. The computer-implemented method in claim 8 wherein at least one of said tag-based text templates drives a format of the data dynamically retrieved from said one or more data sources in response to said request.

11. The computer-implemented method in claim 8 wherein said tag-based text templates include HTML templates.

12. The computer-implemented method in claim 1 wherein said step of processing said request further includes the step of dynamically updating data at said one or more data sources.

13. The computer-implemented method in claim 1 wherein said step of processing said request further includes the step of processing an object handling extension.

14. The computer-implemented method in claim 13 wherein said object handling extension is an OLE extension.

15. A computer-implemented method comprising the steps of:

transferring a request from an HTTP-compliant device to a page server, said page server receiving said request and releasing said HTTP-compliant device to process other requests wherein said transferring step further includes the steps of:

intercepting said request at said HTTP-compliant device and transferring said request to said page server;

processing said request, said processing being performed by said page server while said HTTP-compliant device concurrently processes said other requests; and

dynamically generating a page in response to said request, said page including data dynamically retrieved from one or more data sources.

16. The computer-implemented method in claim 15 wherein said step of transferring said request includes the steps of:

transferring said request from said HTTP-compliant device to a dispatcher; and

dispatching said request to said page server.

17. The computer-implemented method in claim 15 wherein said step of processing said request includes the step of identifying said one or more data sources from which to retrieve said data.

18. The computer-implemented method in claim 15 wherein said step of dynamically generating said page includes the step of dynamically retrieving said data from said one or more data sources.

19. The computer-implemented method in claim 15 wherein said step of processing said request includes the step of said page server maintaining a connection cache to said one or more data sources.

20. The computer-implemented method in claim 15 wherein said step of processing said request includes the step of logging into said one or more data sources.

21. The computer-implemented method in claim 15 wherein said step of dynamically generating said page includes the step of maintaining a page cache containing said page.

22. The computer-implemented method in claim 15 wherein said page server includes tag-based text templates for configuring said page.

23. The computer-implemented method in claim 22 wherein said step of processing said request further includes the step of inserting said dynamically retrieved data from said one or more data sources into said tag-based text templates.

24. The computer-implemented method in claim 22 wherein at least one of said tag-based text templates drives a format of the data dynamically retrieved from said one or more data sources in response to said request.

25. The computer-implemented method in claim 22 wherein said tag-based text templates include HTML templates.

26. The computer-implemented method in claim 15 wherein said step of processing said request further includes the step of dynamically updating data at said one or more data sources.

27. The computer-implemented method in claim 15 wherein said step of processing said request further includes the step of processing an object handling extension.

28. The computer-implemented method in claim 27 wherein said object handling extension is an OLE extension.

29. A computer-implemented method comprising the steps of:

transferring a request from an HTTP-compliant device to a dispatcher;

maintaining dynamic information regarding data sources a given page server may access;

dispatching said request to an appropriate page server based on said request and based on said dynamic information, said page server receiving said request and

releasing said HTTP-compliant device to process other requests;

processing said request, said processing being performed by said page server while said HTTP-compliant device concurrently processes said other requests; and

dynamically generating a page in response to said request, said page including data dynamically retrieved from one or more data sources.

* * * * *

Exhibit D

FILE COPY

JENNER&BLOCK

September 10, 2007

Jenner & Block LLP
330 N. Wabash Avenue
Chicago, IL 60611
Tel 312-222-9350
www.jenner.com

Chicago
Dallas
New York
Washington, DC

VIA EMAIL

David L. Doyle, Esq.
Vedder, Price, Kaufman & Kammholz, P.C.
222 North LaSalle Street
Chicago, Illinois 60601

Patrick L. Patras
Tel 312 923-2945
Fax 312 840-7345
ppatras@jenner.com

Re: *QuinStreet, Inc. v. epicRealm Licensing, LP* (C.A. No. 06-cv-495 (SLR))

Dear Dave:

This is a follow-up to our meet and confer session held at your office on Wednesday, September 5, 2007. It appears that we reached an agreement on the outstanding protective order issues, subject to your further consideration; you agreed that you would get back to us.

We did not reach an agreement on QuinStreet's objections to epicRealm's document requests. There were a number of issues here, but two central ones. *First*, QuinStreet continued to assert that it could withhold documents in the absence of agreement on a protective order. This position is without merit given that Delaware Local Rule 26.2 provides for an "outside counsel's eyes only" protective order in the absence of some other protective order to ensure that document production goes forward on a timely basis, and is not delayed while a protective order is being negotiated. Hopefully, this will be mooted by an agreement on a protective order, but if it is not, we intend to raise the issue with the Court.

Second, we could not agree on the scope of QuinStreet's document production. While there were a number of issues, two were central to the parties' disagreement. The first was that QuinStreet contended that it had no obligation to produce documents for the requested products and services in the absence of epicRealm's infringement contentions. Given that the Court's scheduling order provides for a mechanism (interrogatories) for obtaining contentions and that QuinStreet had not yet served any interrogatories on epicRealm (contention or otherwise), QuinStreet's argument has no merit. QuinStreet has since served contention interrogatories and epicRealm will timely respond.

The second issue regarding the scope of QuinStreet's document production was that QuinStreet objected to the breadth of epicRealm's document requests. You asserted that epicRealm requested virtually every document in QuinStreet's possession, but that is certainly not the case. The scope of epicRealm's requests is defined by the requests themselves but more importantly by the definition of "QuinStreet's Web Products and Services" in ¶18, pp. 3-4. While QuinStreet contended that epicRealm's definition was too broad, you provided no concrete alternative. However, we offer a compromise below.

We also asked if QuinStreet would agree that (at a minimum) QuinStreet produce documents relating to the four QuinStreet "platforms" that QuinStreet identified as being capable of creating dynamic web pages, and on which QuinStreet brought suit seeking a declaration of noninfringement with respect to epicRealm's patents. We called to your attention QuinStreet's assertion in its complaint for declaratory judgment:

PRAYER

> WHEREFORE, QuinStreet requests entry of judgment in its favor and against epicRealm as follows:
>
> a. Declaring that QuinStreet and its software products that are used in conjunction with the delivery of dynamic web pages do not infringe, literally or under the doctrine of equivalents, either directly, indirectly or willfully, any valid and enforceable claim of the epicRealm patents.

(Complaint at p. 7.) We also call attention to QuinStreet's response to Interrogatory No. 1:

> [Q]uinStreet currently utilizes and has in the past utilized the following web site software platforms to power the web servers on which the various web sites reside: Apache (stand alone); Apache/Tomcat (JBoss); Apache/Weblogic; and Microsoft IIS. QuinStreet seeks a declaration that none of these software platforms infringe epicRealm's patents.

(Plaintiff QuinStreet, Inc.'s Responses to Defendant's First Set of Interrogatories at p. 5.) Clearly, by QuinStreet's own admission, documents relating to the above four identified QuinStreet platforms for producing dynamic web pages are relevant or reasonably calculated to lead to the discovery of admissible evidence. Relevance here is clearly established by the fact that QuinStreet's act of seeking declaratory relief of noninfringement can only mean that those platforms are at risk of infringing the epicRealm patents-in-suit. Thus, QuinStreet has no basis for QuinStreet's failure to produce documents on the very four platforms for which QuinStreet seeks a declaration of noninfringement.

In order to reach a compromise with respect to the scope of document production, epicRealm proposes the following narrower definition of QuinStreet's Web Services and Products that are the subject of epicRealm's document requests (and proposes replacing the definition of QuinStreet's Web Products and Services in ¶18 of epicRealm's document requests):

> 1. The four QuinStreet platforms identified above, *i.e.*, Apache (stand alone); Apache/Tomcat (JBoss); Apache/Weblogic; and Microsoft IIS, and any other product providing comparable functionality with respect to the creation, generation, management

or delivery of dynamic web pages (*see* QuinStreet's response to Interrogatory No. 1);

2. QuinStreet's systems for responding to requests for dynamic web pages (*see* ¶16 of QuinStreet's Complaint);

3. QuinStreet's software products that are used in conjunction with the delivery of dynamic web pages (*see* QuinStreet's Prayer for Relief);

4. All current and past versions of QuinStreet's Corporate Business Management suite, Distributor Business Management suite, and Distributor Sales Management suite (*see, e.g.*, www.quinstreetdss.com);

5. Any database software capable of creating, generating, managing or delivering dynamic web pages; and

6. Any products for use in conjunction with any product or service identified in items (1) - (5).

As you can see, we have considerably narrowed epicRealm's requests. As always, we welcome any proposal that QuinStreet may have. After you have had a chance to review our proposal, let's have another meet and confer.

Very truly yours,

Patrick L. Patras

cc: George S. Bosy, Esq.
Benjamin J. Bradford, Esq.